

Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty

NICOLAS MEULEAU

nm@cs.brown.edu

*Computer Science Department, Box 1910
Brown University, Providence, RI 02912, USA.*

PAUL BOURGINE

bourgine@poly.polytechnique.fr

*Ecole Polytechnique, CREA,
Route de Saclay, F-91128 Palaiseau cedex, France.*

Editor: Sridhar Mahadevan

Abstract. This paper presents an action selection technique for reinforcement learning in stationary Markovian environments. This technique may be used in direct algorithms such as Q-learning, or in indirect algorithms such as adaptive dynamic programming. It is based on two principles. The first is to define a local measure of the uncertainty using the theory of bandit problems. We show that such a measure suffers from several drawbacks. In particular, a direct application of it leads to algorithms of low quality that can be easily misled by particular configurations of the environment. The second basic principle was introduced to eliminate this drawback. It consists of assimilating the local measures of uncertainty to rewards, and back-propagating them with the dynamic programming or temporal difference mechanisms. This allows reproducing global-scale reasoning about the uncertainty, using only local measures of it. Numerical simulations clearly show the efficiency of these propositions.

Keywords: reinforcement learning, exploration vs. exploitation dilemma, Markov decision processes, bandit problems.

1. Introduction

Although it recently proved to be an alternative to classical dynamic programming (DP) based techniques for high dimensional problems, reinforcement learning (RL) is primarily concerned with the adaptive optimization of imperfectly modeled dynamic decision problems (Sutton et al. 1991; Sutton 1992; Mahadevan and Kaelbling 1996; Kaelbling 1996; Kaelbling et al. 1996). Classically, the definition of a reinforcement algorithm supposes the choice of two basic components:

- a technique for calculating and storing the estimated value of each state-action pair,
- a rule for selecting actions.

The problem of action selection is not trivial, since always choosing the estimated best actions often leads the learner to converge on a sub-optimal policy (Kumar

1985). The environment must be explored by sometimes performing estimated non-optimal actions. There is thus a constant dilemma between two contradictory objectives:

- *exploitation* (of past experience), that is, maximization of expected reward, knowing that this cannot be done with certainty because one only possess estimates of some variables;
- *exploration* of the environment in order to make estimates more accurate or timely, by the choice of the least known actions.

This paper proposes a technique to deal with the exploration vs. exploitation dilemma during the adaptive optimization of stationary Markov decision processes (MDPs). This problem already has been addressed in many adaptive control (Martin 1967; Kumar and Becker 1982; Kumar and Lin 1982; Sato et al. 1985, 1988, 1990) and RL (Thrun 1992a; Kaelbling 1993 chap. 9; Fiechter 1994) studies. However, there seems to be a lack of efficient solutions for multi-state decision models as MDPs (Kaelbling 1996), as opposed to bandit problems that represent the single-state case.

Our technique may be implemented in direct (model-free) algorithms such as Watkins' Q-learning (1989), and in indirect (model-based) algorithms such as adaptive dynamic programming (Barto et al. 1991, 1995). It is based on two principles:

1. to define local measures of the uncertainty in the form of *exploration bonuses*, using the theory of bandit problems;
2. to add these bonuses to reward and back-propagate them with the dynamic programming (DP) or temporal difference (TD) mechanisms.

The use of local measures of uncertainty is convenient, but it is dangerous. Without prudence, algorithms can be misled by some particular structures of the environment. This is what happens in most current applications (Thrun 1992a; Kaelbling 1993 chap. 9). The back-propagation of exploration bonuses is an effective way to avoid this drawback. It has already been used by Sutton (1990) to deal with non-stationarity in an indirect algorithm, and mentioned by Thrun (1992a, 1992b). Here we adapt it to the stationary Markovian case, and to algorithms which are exclusively direct, such as Q-learning.

The paper is organized as follows: section 2 sets up the framework of MDPs with unknown transition probabilities. It describes briefly the most popular adaptive algorithms for these problems, and the origins of the exploration versus exploitation dilemma.

Section 3 presents the first main feature of our technique: the local measures of uncertainty used for each state-action pair, based on the theory of bandit problems. We first describe previous results on bandit problems, and then propose a unifying notation for several Bayesian and non-Bayesian approaches. Then we consider using these results to derive solutions for multi-state MDPs with unknown probabilities. We stress several drawbacks inherent to this approach, and show that even if it

seems to be a reasonable solution, such an approach cannot be supported by exact mathematical foundations. In the end of section 3, we describe a first series of algorithms, and show what kind of environment may mislead them.

So far, we have done nothing more than generalize, extend and analyze a previous technique described by Kaelbling (1993). Section 4 presents the second main features of our technique, and the main original contribution of this paper. Here, we explain that adding the exploration bonuses defined previously to rewards, and back-propagating them with a DP or TD mechanism, allows us to overcome the drawbacks of the previous algorithms. Using this idea, we propose a second series of algorithms that should not be misled by the environments in which the first fail. In the end of the section, we try to generalize this technique to a wider class of algorithms.

In section 5, we demonstrate the efficiency of our algorithms using extensive numerical simulations.

2. The Exploration vs. Exploitation Dilemma: the Stationary Markovian Case

A key assumption underlying much reinforcement learning research is that the interaction between an agent and its environment may be modeled as a Markov decision process (MDP) (Howard 1960; Puterman 1994).

2.1. Markov Decision Processes

A finite MDP S is defined as $S = (X, A, P, R, \gamma)$ where $X = \{x_i\}$ is the (finite) state space, $A = \{a_k\}$ is the (finite) action (or decision) space¹, $P = [p_{ij}^k]$ is the transition matrix:

$$p_{ij}^k \stackrel{def}{=} \Pr(x(t+1) = x_j | x(t) = x_i, a(t) = a_k) \quad (\forall t),$$

$R = [r_{ij}^k]$ is the reward matrix: r_{ij}^k is the (deterministic) reward received by the system each time that in x_i it chooses action a_k and then goes to x_j , and $\gamma \in [0, 1)$ is the discount factor: one unit of reward received at time $t+1$ is worth γ unit of reward received at time t . S is stationary because the transition probabilities are the same for all time t .

The strategy that maximizes the expected discounted reward

$$\mathbb{E} \left(\sum_{t=0}^{\infty} \gamma^t \cdot r(t) \right) = \lim_{T \rightarrow \infty} \mathbb{E} \left(\sum_{t=0}^T \gamma^t \cdot r(t) \right), \quad (1)$$

is a mapping $\mu^* : X \rightarrow A$ defined by solving Bellman's fundamental equation

$$v_i = \max_k \left[q_i^k + \gamma \sum_j p_{ij}^k \cdot v_j \right] \quad (\forall i) \quad (2)$$

or its equivalent form

$$v_i^k = q_i^k + \gamma \sum_j p_{ij}^k \cdot \max_l v_j^l \quad (\forall (i, k)), \quad (3)$$

where

$$q_i^k \stackrel{def}{=} \sum_j p_{ij}^k \cdot r_{ij}^k \quad (\forall (i, k)) \quad (4)$$

is the expected immediate reward if the systems executes action a_k in state x_i ; v_i is the expected discounted reward (1) if the system starts in x_i and always follows an optimal policy; and v_i^k is the expected discounted reward (1) if the systems starts in x_i , executes action a_k first, and then always follows an optimal policy. Then we have

$$\mu^*(x_i) = \arg \max_k \left[q_i^k + \gamma \sum_j p_{ij}^k \cdot v_j^k \right] = \arg \max_k v_i^k \quad (\forall i). \quad (5)$$

Bellman's system (3) may be solved by asynchronous DP that calculates successive approximations of v_i^k by iterating

$$V_i^k \leftarrow q_i^k + \gamma \sum_j p_{ij}^k \cdot \max_l V_j^l, \quad (6)$$

for different (i, k) . Such an operation is called a ‘back-up’ of the state-action pair (i, k) . Provided that each (i, k) is backed up an infinite number of times, but whatever the order in which they are selected, we have V_i^k converges to v_i^k for all (i, k) (see, for example, Bertsekas 1982; Puterman 1994).

2.2. Adaptive Optimization

RL is well suited to situations where there is significant uncertainty about some parameters of the model. In the case of MDPs, one possibility is that the transition matrix P , and sometimes the reward matrix R , are initially unknown².

The typical solution proposed by (non-Bayesian) adaptive optimal control is the algorithm called adaptive DP (ADP). First, it tries each action once in each state, so that the maximum likelihood estimate (MLE) of P , $\tilde{P} = [\tilde{p}_{ij}^k]$, is defined by

$$\tilde{p}_{ij}^k = \frac{n_{ij}^k}{n_i^k} \quad (\forall (i, j, k)), \quad (7)$$

where n_{ij}^k is the number of transitions from state x_i to state x_j due to action a_k , observed since time 0, and $n_i^k = \sum_j n_{ij}^k$ is the number of executions of a_k in x_i

since time 0. Note that, at this point, the reward r_{ij}^k is known with certainty for all triple (i, j, k) such that $\tilde{p}_{ij}^k > 0$. The MLE is then used to determine the certainty-equivalent optimal policy, i.e. the policy that would be optimal if \tilde{P} were the real value of the unknown parameter P . This is done by DP that executes back-ups according to

$$V_i^k \leftarrow \tilde{q}_i^k + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l V_j^l, \quad (8)$$

where

$$\tilde{q}_i^k \stackrel{def}{=} \sum_j \tilde{p}_{ij}^k \cdot r_{ij}^k \quad (9)$$

is the estimated value of q_i^k , for all (i, k) . Provided that each action is tried in each state an infinite number of times, the MLE converges to the true value of the unknown parameter, thus ensuring that the policy calculated by successive applications of (8) converges to the optimal policy (cf., e.g., Barto et al. 1991, 1995).

ADP is called an indirect algorithm because it uses a “model” \tilde{P} of the problem, and calculates the estimated optimal policy starting from this estimate in the same way as in the non-adaptive case. On the contrary, RL also proposes direct algorithms that estimate the value of the different state-action pairs without an explicit model of the unknown parameters.

The typical direct algorithm for MDP with unknown transition probabilities and rewards is Watkins’ Q-learning (QL) (1989). This algorithm uses a set of Q-values $\{Q_i^k\}$ to approximate the solutions v_i^k of Bellman’s equation (3). Each time that the execution of action a_k in state x_i leads to state x_j (we say that the transition (i, j, k) occurs or is observed), the value Q_i^k is updated by the amount

$$\Delta Q_i^k = \alpha(n_i^k) \left[r_{ij}^k + \gamma \cdot \max_l Q_j^l - Q_i^k \right], \quad (10)$$

where $\alpha(n_i^k) \in (0, 1)$ is the learning rate. Watkins and Dayan (1992) and Tsitsiklis (1994) proved that if each action is tried an infinite number of times in each state, and if the learning rate $\alpha(n)$ satisfies

$$\sum_{n=0}^{\infty} \alpha(n) = +\infty \quad \text{and} \quad \sum_{n=0}^{\infty} \alpha(n)^2 < +\infty, \quad (11)$$

then, with probability 1, $Q_i^k \rightarrow v_i^k$ as time tends to infinity for all (i, k) .

2.3. Exploration vs. Exploration Dilemma

We see that the convergence to the optimal policy is ensured for both adaptive algorithms (ADP and QL), but it requires that each action be tried infinitely often

in each state. In practice, this is impossible to achieve because we cannot wait an infinite time before starting the optimization. Therefore, at each time t , we have to decide between continuing to sample to ensure convergence (exploration), and following the estimated optimal policy (exploitation).

There have been many propositions for dealing with this problem (Martin 1967, Sato et al. 1985, 1988, 1990, Thrun 1992a; Kaelbling 1993, Fiechter 1994). An intuitive way is to measure the uncertainty attached to each action, and to use these measures and the actions' estimated quality to take decisions.

3. Local Measures of Uncertainty Based on the Theory of Bandit Problems

As other researchers experimented previously (e.g. Kaelbling 1993), we use the theory of bandit problems to define measures of the uncertainty attached to each state-action pair. The origin of this approach is an attempt to distribute the adaptive optimization by considering an independent bandit problem for each state of the original problem. In this work, we use results pertaining to normal bandit problems.

3.1. Normal Bandit Problems

3.1.1. Definition

The exploration vs. exploitation dilemma is very often associated with the K -armed bandit problem where it appears in its simplest form (Kaelbling et al. 1996). To understand this problem (and the origins of its name), we must imagine several slot-machines (or “one-armed-bandits”) side-by-side. A single play of a machine costs the same price whatever the machine. Therefore, everything happens as if there was a single machine with several levers (or “arms”), the player choosing one of them after inserting a coin. We speak then of K -armed bandits.

Each arm delivers a random reward. Because there are initially K independent machines, the rewards delivered by different arms are independent random variables (this is a fundamental hypothesis). Some arms are better than others (i.e., give higher expected rewards), but usually we do not know which is the best.

Except if the arm statistics are completely known (no exploration needed), or if they are completely unknown (no exploitation possible), we face the exploration vs. exploitation dilemma when we play a multi-armed bandit. Exploitation consists of choosing the estimated best arm, and exploration is the choice of another arm, for the sake of checking that estimations are correct, or for making them more accurate.

Each bandit problem is a special case characterized by what is known of the probability distributions on the rewards delivered by each arm. If one knows these laws sufficiently well—which supposes a sufficient knowledge of the uncertainty attached to them—a direct calculation of the solution (according to some optimality

criterion) is possible. The Berry and Fristedt book (1985) provides an extensive bibliography on the subject.

Hereafter, we will use results developed for bandit problems where each arm k delivers the random reward ρ_k that follows a normal (Laplace-Gauss) distribution $N(m_k, \sigma_k^2)$ with mean m_k and variance σ_k^2 . We will consider two cases:

- m_k unknown and σ_k^2 known, for all k ,
- m_k and σ_k^2 are unknown, for all k .

3.1.2. Interval estimation

A non-Bayesian solution to this problem was proposed by Kaelbling (1993). This algorithm, called ‘Interval Estimation’ (IE), consists in always choosing the arm that maximizes the upper-bound ub_k of a $100(1 - \theta)\%$ confidence interval of ρ_k , for some confidence coefficient $\theta \in (0, 1)$.³ We recall that I_θ^k is a $100(1 - \theta)\%$ confidence interval of ρ_k if and only if

$$\Pr(m_k \in I_\theta^k) = 1 - \theta. \quad (12)$$

It can be demonstrated quite easily that, if ρ_k follows a normal distribution with unknown mean m_k and known variance σ_k^2 , then we have

$$ub_k = \bar{\rho}_k + \sigma_k \frac{z_{\theta/2}}{\sqrt{n_k}} \quad (n_k \geq 1) \quad (13)$$

where

$$\bar{\rho}_k \stackrel{def}{=} \frac{\sum \rho_k}{n_k} \quad (n_k \geq 1) \quad (14)$$

is the sample mean of n_k observations of ρ_k , and $z_{\theta/2}$ is the upper $100(\theta/2)\%$ point of the unit normal distributions, i.e., the value $z > 0$ such that $\Pr(N(0, 1) < z) = 1 - \theta/2$.

Another basic result of statistics is that if both the mean and variance of ρ_k are unknown, then

$$ub_k = \bar{\rho}_k + s_k \frac{t_{\theta/2}^{n_k-1}}{\sqrt{n_k}} \quad (n_k \geq 2) \quad (15)$$

where

$$s_k \stackrel{def}{=} \sqrt{\frac{\sum (\rho_k - \bar{\rho}_k)^2}{n_k - 1}} = \sqrt{\frac{n_k \sum \rho_k^2 - (\sum \rho_k)^2}{n_k (n_k - 1)}} \quad (n_k \geq 2) \quad (16)$$

is the sample standard deviation of n_k observations of ρ_k , and $t_{\theta/2}^{n_k-1}$ is Student’s t -function at confidence level $\theta/2$ and with $(n_k - 1)$ degrees of freedom.

In practice, IE first tries each arm once if the variances are known, and twice otherwise. Then, it uses (13) or (15) to calculate the upper bound ub_k associated with each arm. The solution is then to choose, at each time t , an arm that maximizes ub_k , while updating this value each time that a new observation is available. The values of $z_{\theta/2}$ in (13) and of $t_{\theta/2}^{n_k-1}$ in (15) are obtained by consulting tables available in any general statistics book (e.g., Larsen and Marx 1986; Snedecor and Cochran 1989).

3.1.3. Gittins' indices

The Bayesian solution to bandit problems is always associated with powerful mathematical tools known as Gittins' indices (or dynamic allocation indices) (Gittins 1989). The use of Gittins' indices spreads beyond the scope of bandit problems, but bandit problems constitute their primary field of application.

Bayesian adaptive control aims at maximizing the expected discounted reward

$$\mathbb{E} \left(\sum_{t=0}^{\infty} g^t \cdot \rho(t) \right), \quad (17)$$

where $\rho(t)$ is the reward received at time t (which only depends on the arm chosen at this time), and $g \in [0, 1)$ is a discount factor⁴. The expectation in this equation is calculated relative not only to the stochastic behavior of the model, but also to a certain probability distribution on the unknown parameters (the initial belief). In this framework, the Gittins' index is a value ν_k attached to each arm k , and such that it is always optimal (with respect to (17)) to choose an arm that maximizes ν_k .

According to Gittins (1989, section 7.2 and 7.3):

- if ρ_k follows $N(m_k, \sigma_k)$ with mean m_k unknown and uniformly distributed over the real line, and known variance σ_k^2 , then

$$\nu_k = \bar{\rho}_k + \sigma_k \cdot \nu_g(0, n_k) \stackrel{def}{=} \nu_g(\bar{\rho}_k, n_k) \quad (n_k \geq 1), \quad (18)$$

where $\bar{\rho}_k$ is defined as in (14) and $\nu_g(0, n) \geq 0$ is the index of a normal arm with unknown and uniformly distributed mean, with variance known and equal to 1, and after n observations with sample mean 0 (i.e. the value that the index should have if $\bar{\rho}_k = 0$ and $\sigma_k = 1$);

- if both the mean and variance of ρ_k are unknown and uniformly distributed over the real line, then

$$\nu_k = \bar{\rho}_k + s_k \cdot \nu_g(0, 1, n_k) \stackrel{def}{=} \nu_g(\bar{\rho}_k, s_k, n_k) \quad (n_k \geq 2), \quad (19)$$

where s_k is defined as in (16), and $\nu_g(0, 1, n) \geq 0$ is the index of a normal arm with unknown and uniformly distributed mean and variance, after n observations with sample mean 0 and sample standard deviation 1 (i.e. the value that the index should have if $\bar{\rho}_k = 0$ and $s_k = 1$).

The values of $\nu_g(0, n)$ in (18) and $\nu_g(0, 1, n)$ in (19) may be found in tables provided by Gittins⁵ (1989).

The algorithm derived from these results is very close to IE: first, try each alternative one or two times, depending on whether the variances σ_k^2 are known or not; then use (18) or (19) to determine the next action to perform, by referring to tables when necessary, and by updating the statistics $\bar{\rho}_k$, s_k and n_k each time that an outcome of arm k is observed.

3.1.4. Conclusion: exploration bonuses

We see that whatever the nature of the approach (Bayesian or not), the optimal solution to normal bandit problems is always to choose, after some steps, the arm that maximizes the estimated average reward $\bar{\rho}_k$ plus the *exploration bonus* δ_k defined by

$$\delta_k \stackrel{\text{def}}{=} \begin{cases} \sigma_k \cdot \delta_0(n_k) & \text{in the case of known variance,} \\ s_k \cdot \delta_0(n_k) & \text{in the case of unknown variance,} \end{cases} \quad (20)$$

where the *unit exploration bonus* $\delta_0(n)$ is a positive decreasing function of n defined by equations (13), (15), (18) or (19) as

$$\delta_0(n) \stackrel{\text{def}}{=} \begin{cases} z_{\theta/2}/\sqrt{n} & \text{(interval estimation, known variance),} \\ t_{\theta/2}^{n-1}/\sqrt{n} & \text{(interval estimation, unknown variance),} \\ \nu_g(0, n) & \text{(Gittins' indices, known variance),} \\ \nu_g(0, 1, n) & \text{(Gittins' indices, unknown variance).} \end{cases} \quad (21)$$

Practically, this value is calculated easily starting from tables.

The exploration bonus δ_k represents the maximum amount (of reward or utility) that one is willing to pay for one observation of the output of arm k (fig. 1). It measures the importance of sampling this arm to obtain information rather than simply obtaining the expected reward $\bar{\rho}_k$. In all cases we have $\lim_{n \rightarrow \infty} \delta_0(n) = 0$. This reflects the fact that, when an arm has been tried an infinite number of times, its characteristics are known with certainty and thus, there is no further information to be learned from sampling it.

Because the bonuses are used in additive equations, their order of magnitude is very important and scaling them may dramatically corrupt the behavior they induce. At each time t , the optimal decision does not only depend on the order of preference of the different arms according to the current knowledge, but it also depends on the very value attached to each arm (cf. 4.2.2).

Figures 2 and 3 show plots of the non-Bayesian and Bayesian unit exploration bonuses $\delta_0(n)$, for small values of n . It is striking to observe :

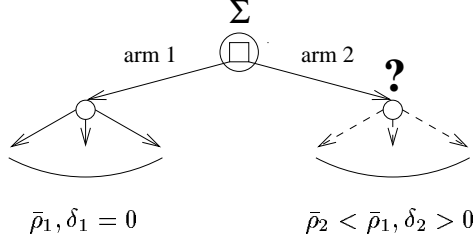


Figure 1. If the system Σ chooses arm 2, its estimated loss is $\bar{\rho}_1 - \bar{\rho}_2 > 0$, but it makes a new observation of arm 2. It will make this choice (and pay this price) as long as $\bar{\rho}_2 + \delta_2 \geq \bar{\rho}_1$ i.e. $\bar{\rho}_1 - \bar{\rho}_2 \leq \delta_2$. Thus, the maximum amount it is willing to pay for an observation of arm 2 is equal to the exploration bonus δ_2 .

- the similarity between graphs 2.a and 3.a and between graphs 2.b and 3.b,
- the rapid decrease of $\delta_0(n)$ during the first steps of the exploration, especially in the case of unknown variance,
- the higher values of non-Bayesian bonuses (fig. 2) than the Bayesian (fig. 3).

With regard to this last point, it is to be noticed that because $\lim_{n \rightarrow \infty} t_{\theta/2}^{n-1} = z_{\theta/2}$, the non-Bayesian bonus is always equivalent to $z_{\theta/2} \cdot n^{-1/2}$ when n tends to infinity. Moreover, according to Gittins we have

$$\lim_{n \rightarrow \infty} \frac{\nu_g(0, 1, n)}{\nu_g(0, n)} = 1^+ \quad (\forall g \in [0, 1]), \quad (22)$$

and $n \cdot \nu_g(0, n)$ always converges to a positive value as n tends to infinity. Therefore, the Bayesian bonus behaves as $K \cdot n^{-1}$ for some $K > 0$ as n tends to infinity. We see that, in general, the non-Bayesian optimal behavior results in more exploration than the Bayesian.

Before ending this section, we notice that all the algorithms presented here are subject to the “sticking problem” (Kaelbling 1993, section 4.2.2): only one arm is played infinitely often, and there is a non-zero probability that it is a non-optimal arm that is selected. This limitation is present in every problem of adaptation to a stochastic process: one can never be sure to have found the best alternative before having tried all of them an infinite number of times. Clearly, this is in contradiction to the necessity eventually to exploit the acquired knowledge. In response to this problem, Sato et al. (1982, 1985, 1988, 1990) have developed the notion of “asymptotic optimality” that consists of trying every alternative infinitely often, while making the frequency of use of the estimated best tend to 1. It is noteworthy that both the Bayesian and the non-Bayesian approaches lead to abandoning every arm except the estimated best, and thus risk making mistakes from time to time.

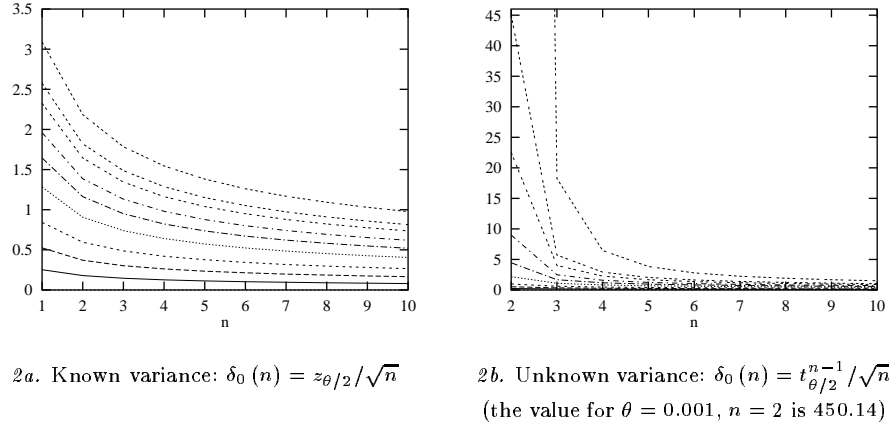


Figure 2. Interval estimation (non-Bayesian) unit exploration bonuses for $\theta = 0.8, 0.6, 0.4, 0.2, 0.1, 0.05, 0.02, 0.01$ and 0.002 (known variance) or 0.001 (unknown variance) (the interval estimation bonus is a decreasing function of θ).

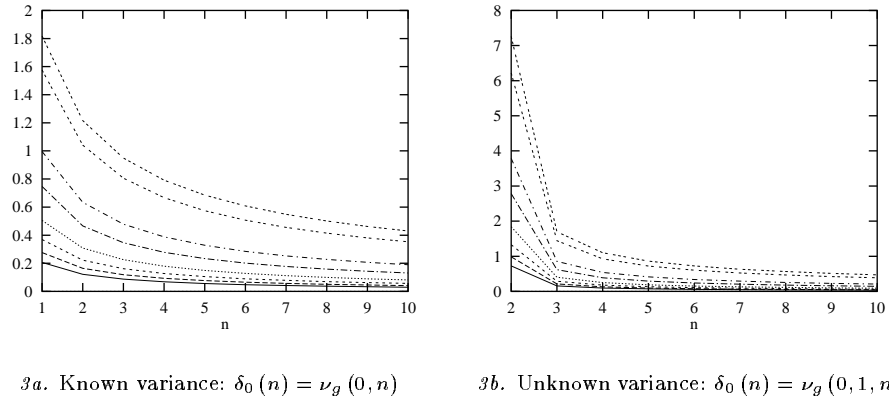


Figure 3. Gittins' (Bayesian) unit exploration bonuses for $g = 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99$ and 0.995 (the Gittins' bonus is an increasing function of g).

3.2. Distributing the Learning Task

Our purpose is to use results from the theory of bandit problems, and particularly equations (20) and (21), to define local measures of the uncertainty for the multi-state problem. For this reason, we examine the possibility of distributing the learning task by representing the problem of adaptation to a multi-state MDP $S = (X, A, P, R, \gamma)$ ($|X| > 1$), as a set of $|X|$ bandits, each with $|A|$ arms.

The bandit problem associated with state x_i will be called B_i . The execution of action a_k in x_i is equivalent to a pull of arm k of B_i . The distributed approach consists in considering a different sub-system Σ_i “playing” the bandit B_i , for all i . The reward earned by Σ_i when it pulls arm k is ρ_i^k , to be specified.

3.2.1. Non-stationarity of each bandit problem

The first thing to do is to define ρ_i^k , the reward earned by sub-system Σ_i when it plays arm k . An obvious solution is to chose the reward received immediately after the execution of a_k in x_i , i.e., ρ_i^k follows the law

$$\rho_i^k = r_{ij}^k \quad \text{with probability } p_{ij}^k. \quad (23)$$

However, in this case, each Σ_i will consider exploitation to consist of maximizing the expected immediate reward q_i^k defined by (4). Thus, the overall system $\Sigma = \bigcup_i \Sigma_i$ is adaptively optimizing the one-step-horizon reward $E(r(0))$, which is often very different from the infinite-horizon reward (1).

If we want to be able to manage the temporal credit assignment problem, the reward received by Σ_i must contain some information about the value of the arrival states. An ideal solution is

$$\rho_i^k = r_{ij}^k + \gamma \cdot v_j \quad \text{with probability } p_{ij}^k. \quad (24)$$

An algorithm based on this law would avoid the drawback of definition (23). Unfortunately, such an algorithm is impossible to implement because we do not know the true values v_i^k , and we only have some time-varying approximations of them called V_i^k or Q_i^k (equations (8) and (10)). Hereafter, we are limited to the definition

$$\rho_i^k = r_{ij}^k + \gamma \cdot \max_l \begin{pmatrix} V_j^l \\ \text{or} \\ Q_j^l \end{pmatrix} \quad \text{with probability } p_{ij}^k, \quad (25)$$

that constitutes a direct adaptation of (24). In this case, the distribution of ρ_i^k varies as the V- or Q-values vary. Therefore, each bandit problem B_i is not stationary, even if the original problem S is. The consequences in imposing a forgetting mechanism for each Σ_i are important, and have already been stressed by Kaelbling (1993, p. 150).

3.2.2. Non-independence of different arms

If ρ_i^k follows (25), then the reward delivered by arm k of bandit B_i depends on the estimated value of each possible arrival state. With both ADP and QL, the value of a state is updated each time this state is visited. Thus, when Σ_i chooses arm k and Σ moves from x_i to x_j , the value of the arrival state x_j is going to be modified

at the next time step. As a consequence, the law followed by ρ_i^k is going to change. Similarly, this is the case for each $\rho_i^{k'}$ with $k' \neq k$, such that (i, j, k') is a possible state transition.

In this way, the choice of an arm changes the distributions associated with other arms. Thus, the fundamental hypothesis of independence of the different arms is not respected, and the local sub-problems are not bandit problems.

3.2.3. *Non-independence of different sub-problems*

In the same way as above, we can show that the choice of arm k in x_i changes the distributions of some $\rho_{i'}^{k'}$ with $i' \neq i$. The consequence is that the different sub-problems are not independent.

3.2.4. *Conclusion*

We see that a set of independent bandit problems constitutes an approximate and false model of the original problem. Therefore, a decentralized approach based on bandit problems is essentially heuristic, and may not have an exact theoretical foundation. However, this should not discourage empirical studies of this approach.

3.3. Basic Algorithms

One of simplest ways to design algorithms is to assume that Σ_i approximates ρ_i^k by a normal distribution, for all (i, k) . Then we can use the results presented in section 3.1.

3.3.1. *Outline*

Depending on whether we use a model of the process or not, V- or Q-values are kept up to date using (8) or (10). Each time that transition (i, j, k) occurs, Σ_i receives the reward

$$\rho_i^k = r_i^k + \gamma \cdot \max_l \left(\begin{array}{c} V_j^l \\ \text{or} \\ Q_j^l \end{array} \right), \quad (26)$$

then it updates:

- n_i^k , the number of pulls of arm k ,
- $\bar{\rho}_i^k$, the mean of the rewards received from arm k , defined as (14),
- s_i^k , the sampled standard deviation of these rewards, defined as (16).

From these statistics, it calculates

$$\begin{pmatrix} ub_i^k \\ \text{or} \\ \nu_i^k \end{pmatrix} = \bar{\rho}_i^k + \delta_i^k \quad (\forall (i, k)), \quad (27)$$

where

$$\delta_i^k = s_i^k \begin{pmatrix} t_{\theta/2}^{n_i^k-1} / \sqrt{n_i^k} \\ \text{or} \\ \nu_g(0, 1, n_i^k) \end{pmatrix} = s_i^k \cdot \delta_0(n_i^k) \quad (\forall (i, k)), \quad (28)$$

for a given confidence coefficient θ or discount factor g . The value δ_i^k represents a local measure of uncertainty about the infinite-horizon rewards. It is an exploration bonus adapted to the multi-state problem.

The non-stationarity of each B_i is managed by simulating a forgetting mechanism in each Σ_i . In our applications, we calculate the statistics n_i^k , $\bar{\rho}_i^k$ and s_i^k , and the exploration bonuses δ_i^k , on equally-weighted sliding windows of length L_{win} (i.e. using only the last L_{win} observations of ρ_i^k).

Σ_i first tries every arm twice, then it always chooses an arm that maximizes ub_i^k or ν_i^k .

3.3.2. Fundamental equation

When ρ_i^k follows (25) we have

$$\bar{\rho}_i^k \simeq \mathbb{E}(\rho_i^k) = q_{ij}^k + \gamma \sum_j p_{ij}^k \cdot \max_l \begin{pmatrix} V_j^l \\ \text{or} \\ Q_j^l \end{pmatrix} \simeq \begin{pmatrix} V_i^k \\ \text{or} \\ Q_i^k \end{pmatrix} \quad (\forall (i, k)). \quad (29)$$

Thus, the natural criterion (27) may be approximated by

$$N_i^k \stackrel{def}{=} \begin{pmatrix} V_i^k \\ \text{or} \\ Q_i^k \end{pmatrix} + \delta_i^k \quad (\forall (i, k)). \quad (30)$$

This is a direct way to design algorithms where the statistic $\bar{\rho}_i^k$ is kept in memory only because it is needed to calculate s_i^k . Upon seeing that the variables V_i^k and Q_i^k are better estimates of the quality of the state-action pair (i, k) than the mean $\bar{\rho}_i^k$ (especially if $\bar{\rho}_i^k$ is calculated on a sliding window), we prefer to use equation (30) instead of the original (27). Moreover, we consider that this equation is characteristic of the algorithms developed in this section (as opposed to those defined in section 4).

- 1. Initialize:**
 - set the counters to 0: $n_{ij}^k = n_i^k = 0$ for all (i, j, k) ,
 - initialize the sliding window of each state action pair (i, k) ,
 - initialize the main variables: $V_i^k = 0$ (ADP) or $Q_i^k = 0$ (QL) for all (i, k) .
 - 2. Choose an action** (the current state is x_i):
 - if there exists k' such that $n_i^{k'} < 2$, then choose $k = k'$,
 - otherwise choose k which maximizes (30).
 - 3. Execute action** a_k , observe arrival state x_j .
 - 4. Update variables:**
 - increment the counters n_i^k and n_{ij}^k ,
 - update the estimate \tilde{p}_{ij}^k with (7), if using ADP,
 - update the main variables:
 - ADP: update $V_{i'}^{k'}$ with (8) and (9) for different (i', k') such that $n_{i'}^{k'} > 0$,
 - QL: update Q_i^k with (10),
 - add an observation to the sliding window of state-action pair (i, k) with (26) (variance-based) or (32) (error-based),
 - if $n_i^k > 1$:
 - update the statistics $\bar{\rho}_i^k$ and s_i^k with (14) and (16),
 - update the exploration bonus δ_i^k with (28).
 - 5. Increment time**, return to 2.

Figure 4. Outline of “–” algorithms’ variance- and error-based variants (ADP and QL).

3.3.3. Algorithms

Our algorithms are defined by the following set of formulas:

- the equations of ADP or QL: (7), (8) and (9); or (10) and (11);
- the definition of the reward ρ_i^k introduced in the sliding windows: (26);
- the definition of the statistics $\bar{\rho}_i^k$ and s_i^k : (14) and (16);
- the definition of the exploration bonus: (28);
- the definition of the criterion of choice: (30).

Figure 4 provides an outline of the algorithms. We call these algorithms IDP–, IEDP–, IQL– and IEQL– where :

- I stands for “indices” and denotes the choice of the Bayesian framework, while IE denotes the choice of a non-Bayesian algorithm,
- DP and QL stand for ADP and Q-learning,
- “–” distinguishes these algorithms from those proposed in section 4.

IEQL– is very close to Kaelbling’s algorithm (1993, chap. 9). The two algorithms differ only on the following points:

- we use equation (30) instead of the natural criterion (27) used by Kaelbling,

- the mechanisms used to forget observations at the level of each Σ_i are not the same: Kaelbling uses a geometrical decay of statistics whereas we use sliding windows.

3.4. Variants

Thrun (1992a, 1992b) has proposed classifying the techniques of action selection according to the information used. The first distinction he introduces is between:

- *undirected techniques* that do not use any “exploration-specific” knowledge about the learning process,
- *directed techniques* that remember knowledge about the learning process and use it to direct exploration.

Undirected techniques proceed by drawing at random the executed action, in a way that favors the estimated best. They are distinguished from each other by the probability distribution used for these drawings. The two most used are the semi-uniform (or ϵ -greedy) distribution (e.g. Watkins 1989; Whitehead and Ballard 1991) and the Boltzmann law (e.g. Watkins 1989; Barto et al. 1991, 1995; Kaelbling et al. 1996).

Most directed techniques are heuristic and are not theoretically motivated. However, they use the notion of an exploration bonus. The criterion they maximize may be put in the form

$$N_i^k = \left(\begin{array}{c} V_i^k \\ \text{or} \\ Q_i^k \end{array} \right) + \delta_i^k + \dots \quad (31)$$

where δ_i^k is an exploration bonus that measures the uncertainty in a particular way, and the dots indicate that several bonuses of different nature may be added.

Thrun proposes to classify the directed techniques according to the information used to select the actions. He distinguishes:

counter-based techniques that keep up-to-date counters n_i^k of the number of times that each state-action pair (i, k) has been tried, and use them during the choice of the actions (e.g., Sato et al. (1985, 1988, 1991) asymptotically optimal algorithms). When two actions have the same estimated value, they choose the one that has been tried the least often previously;

error-based techniques that use the measured (or predicted) variations of the variables V_i^k or Q_i^k during their last (or next) update(s), in the rule of selection of the actions (e.g., Moore 1990; Schmidhuber 1991a; Thrun and Moller 1991, 1992). In general, these techniques prefer the states or actions whose estimated quality varied the most in the past, or is predicted to vary the most in the future;

recency-based techniques that deal with non-stationary problems by keeping in memory the date of the last trial of each state-action pair, and choosing preferentially the actions that have been tried the least recently (e.g. Sutton 1990, 1991a).

Some techniques are mixed-techniques, i.e., they use pieces of information of different nature. This is achieved by adding several different exploration bonuses in (31) (e.g. counter-/error-based techniques proposed by Thrun), or by defining mixed exploration bonuses (see below).

3.4.1. Variance-based algorithms

Thrun's lexicon does not cover the case of the algorithms presented in sections 3.1 and 3.3. Because the exploration bonus δ_k depends on n_k (equation (20)), these algorithms are counter-based techniques. However, the counters are not the only information used in the exploration bonuses: the standard deviations σ_k or s_k of the rewards are present as well. When two actions have been tried the same number of times and have the same estimated value, the algorithms choose the one which delivers the most random reward. In this way, they show a preference for risk.

To qualify these algorithms, we introduce the following definition:

variance-based techniques measure the variability of the outcome of different actions in order to calculate of their exploration bonuses.

Thus, the bandit-problem solutions of section 3.1 are variance-/counter-based techniques. The algorithms for multi-state MDPs of section 3.3 (and Kaelbling's algorithm) are also variance-/counter-based techniques: their exploratory behavior depends mainly on counters n_i^k and standard deviation s_i^k of different state-action pairs (i, k) . However, their case is more complicated: due to of the presence of the variable V_j^l or Q_j^l in the right term of (26), they are also, to a smaller extent, error-based techniques. In the following, we will refer to them simply as variance-based IDP, IEDP, IQL and IEQL.

3.4.2. Error-based algorithms

The use of the rule

$$\rho_i^k = \begin{pmatrix} V_i^k \\ \text{or} \\ Q_i^k \end{pmatrix} \quad (32)$$

instead of (26) is another way to define the local reward received by sub-system Σ_i when it pulls arm k . In this way, all that is developed in section 3.2 is still true, and equation (30) is still an approximation of (27). The algorithms using this law measure the variations of the V- or Q-values during their last updates. Thus,

they are error-/counter-based techniques using mixed exploration bonuses. In the following, we will refer to them simply as error-based IDP, IEDP, IQL and IEQL.

3.4.3. Worst-case algorithms

The development of the algorithms presented here was motivated by the following facts. A wide class of problems are characterized by a sparse reward matrix R , i.e. $r_{ij}^k = 0$ for most triples (i, j, k) . In this case, if the V- or Q-values are initialized to 0, then they remain zero during a first stage that may last for a long time⁶. Thus, most of the standard deviations s_i^k and exploration bonuses δ_i^k of the algorithms based on (26) or (32) degenerate to 0 during the first steps of the experience. Therefore, the behavior of variance- and error-based algorithms is greatly degraded, and it may turn to random walk, arbitrary choice of always the same actions, or greedy exploitation.

To avoid this problem, we first note that if

$$r_M \stackrel{def}{=} \max_{i,j,k} r_{ij}^k \quad \text{and} \quad r_m \stackrel{def}{=} \min_{i,j,k} r_{ij}^k, \quad (33)$$

and the initial V- or Q-values satisfy

$$\frac{r_m}{1-\gamma} \leq \left(\begin{array}{c} V_i^k \\ \text{or} \\ Q_i^k \end{array} \right) \leq \frac{r_M}{1-\gamma} \quad (\forall (i, k)), \quad (34)$$

then this equation is satisfied at each time t . Therefore, the standard deviation σ_i^k of a random variable defined by (26) or (32) satisfies

$$\sigma_i^k \leq \frac{r_M - r_m}{2(1-\gamma)} \stackrel{def}{=} \sigma_{max} \quad (\forall (i, k)). \quad (35)$$

We propose to use this result and to suppose that the standard deviation of ρ_i^k is known and equals σ_{max} , for all (i, k) . Then we use the formulas for normal arms with known variance (13) and (18) and define the exploration bonus as

$$\delta_i^k = \sigma_{max} \left(\begin{array}{c} z_{\alpha/2} / \sqrt{n_i^k} \\ \text{or} \\ \nu_g(0, n_i^k) \end{array} \right) = \sigma_{max} \cdot \delta_0(n_i^k) \quad (\forall (i, k)). \quad (36)$$

The statistics $\bar{\rho}_i^k$ and s_i^k are not used anymore, and neither are the sliding widows. The algorithms are then greatly simplified. We call them *worst-case* IDP-, IEDP-, IQL- and IEQL-, because they suppose that the standard deviation is always equal to its biggest (worst) possible value. They are defined by the following set of formulas:

- the equations of ADP or QL: (7), (8) and (9); or (10) and (11);

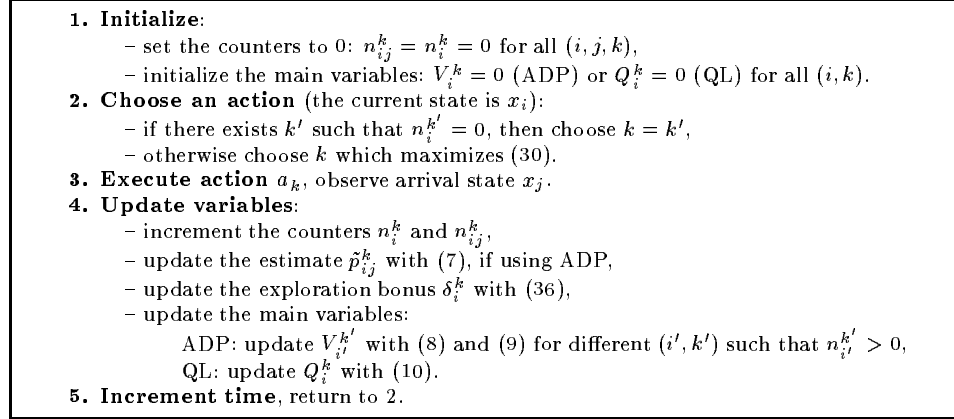


Figure 5. Outline of “–” algorithms’ worst-case variant (ADP and QL).

- the definition of the exploration bonus: (36);
- the definition of the criterion of choice: (30).

Figure 5 provides an outline of the algorithms.

Worst-case algorithms are pure counter-based techniques. They avoid the degeneration of the exploration bonuses that happens with sparse reward matrices. However, they need to know in advance the difference $(r_M - r_m)$, and thus, they need more information than the previous algorithms.

From the optimal control point of view, the rewards represent the objective function and thus they are initially given. From the decision theory point of view, the rewards represent the utility function, also known in advance. In general, we consider that specifying its reward structure to a system is telling it its task. Thus, as long as we are concerned with optimization and problem solving, the rewards are always known in advance.

However, if this is not the case and the difference $(r_M - r_m)$ is initially unknown, then this value must be measured on-line. In this case, a positive constant must be used in place of σ_{max} as long as two different rewards have not been received. The value of this constant has no influence on the algorithm’s behavior.

3.5. Conclusion

As we have shown earlier, the algorithms developed in this section do not rely on rigorously valid theoretical foundations. However, they represent rational propositions supported by common sense arguments.

Each Σ_i uses an optimal solution of single-state problems, exploring more or less each action, and most likely ultimately converging to the estimated best. The

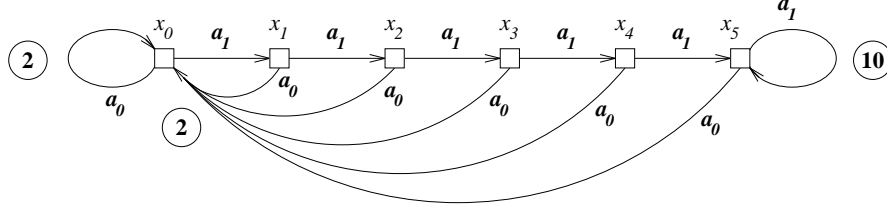


Figure 6. Task 1 ($\gamma = 0.99$) (Meuleau 1996).

aggregation of these local behaviors works well with problems where the different states are visited equally often. However, it is held in check by particular configurations of the environment where some states are visited much more often than others. As a matter of fact, because we consider a set of $|X|$ independent bandit problems, each Σ_i functions within its own concept of time. For Σ_i time elapses only when it plays, and it is frozen otherwise. Thus, ‘-’ algorithms may be misled by problems where some important states are rarely visited.

This is the case of the problem represented in fig. 6. In this deterministic problem, one always has the choice of going to the next index state and winning nothing, or returning to state x_0 and receiving 2. Once arriving in the higher index state x_5 , one may stay there and win 10, or return to x_0 and win 2. With $\gamma = 0.99$, the optimal policy is to choose a_1 everywhere. However, this may appear late in the V- or Q-values, because the reward 10 must be back-propagated along the whole chain of states. Moreover, state x_0 is visited very often. Thus, Σ_0 may become self-confident too early, and start exploitation before the real V- or Q-values are known. It may then converge to the sub-optimal action a_0 and prevent further exploration.

The deterministic problem presented in fig. 7 was designed by Watkins (1989) to be misleading. It functions with the same principle as task 1: Σ_0 may become self-confident before the reward 2 has been back-propagated until x_0 , and then converge on the sub-optimal action a_1 .

To solve this kind of problem, we introduce now the second basic principle of our algorithms.

4. Back-Propagation of Uncertainty

Local measures of uncertainty are convenient, but a direct use of them leads to a local-scale reasoning about information that is insufficient in many multi-state environments. The back-propagation of the exploration bonuses is an easy way to simulate global-scale reasoning about the uncertainty, using only its local measures.

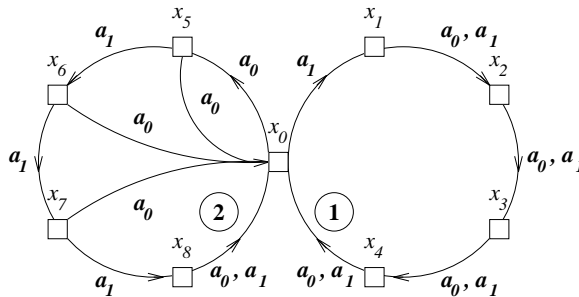


Figure 7. Task 2 ($\gamma = 0.9$) (Watkins 1989).

4.1. Local vs. Global Exploration Policies

We say that the algorithms based on (30) represent local exploration policies, because their decisions do not take into account the uncertainty attached to states other than the current state.

In contrast, we say that an algorithm implements a global exploration policy if it takes into account, during the choice of its actions, the uncertainty attached to states other than the current state. Intuitively, global exploration strategies are such that an observation made in some state x_i may change the decision in other states x_j ($j \neq i$), independently of any change in the V- or Q-values that may result from the observation. An instance of global scale reasoning is presented in fig. 8. These strategies should not be misled by problems such as tasks 1 and 2.

There are a few algorithms able to simulate global exploration policies. To our knowledge, only three techniques are available:

1. Bayesian adaptive control, applied to finite multi-state MDPs by Martin (1967);
2. Feldbaum's (1965) "caution and probing" technique, applied to dynamic programming by Bar-Shalom (1981);
3. Sutton's Dyna-Q+ algorithm (1990).

The first two proceed by introducing the knowledge about the process into the state of the system, which is then called "super-state" or "information-state". Then, applying DP to the reformulated problem simulates global scale reasoning about uncertainty. Although this approach constitutes an elegant theoretical solution, it requires prohibitive amounts of computer time and space, and thus it is not applicable to most real-world problems.

Sutton's Dyna-Q+ algorithm (1990) is the first instance of an algorithm's using the principle of back-propagation of uncertainty. It seems that it is the only realistic solution proposed before ours. We will first present our algorithms, and then show their similarities and differences with Dyna-Q+ (section 4.4).

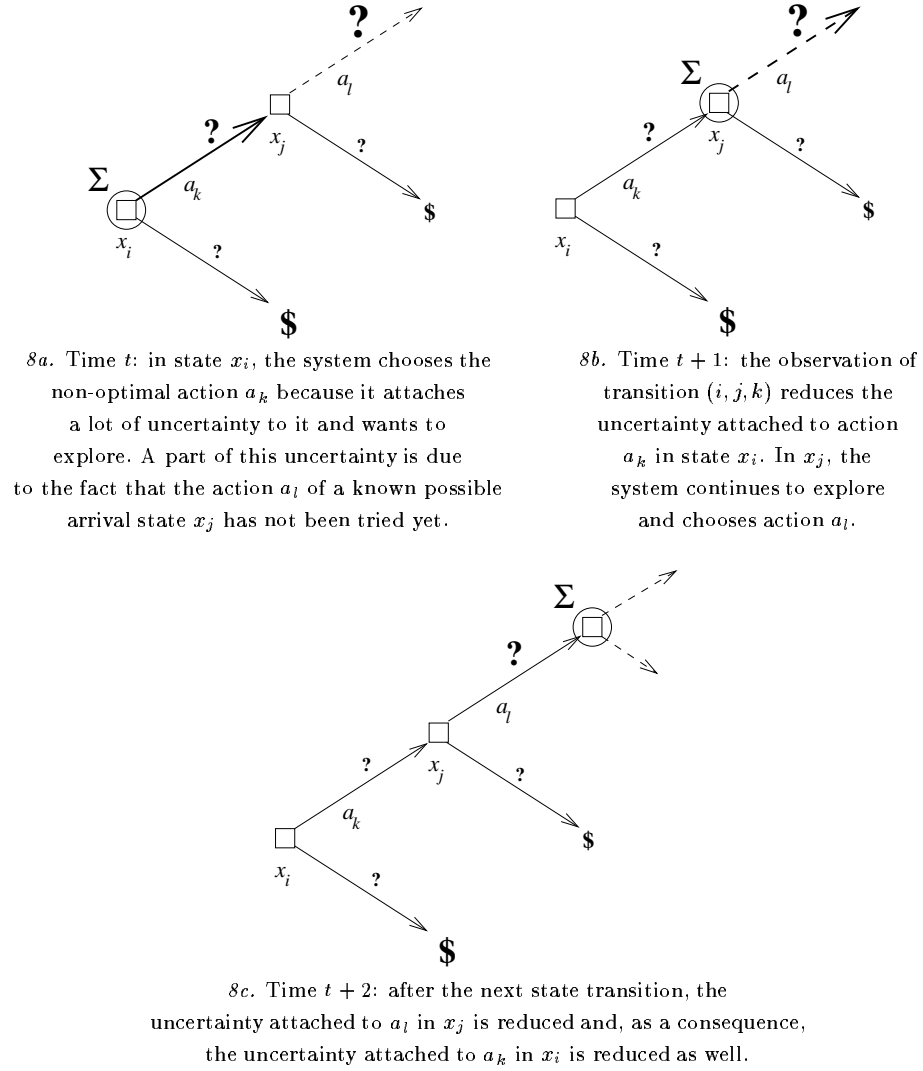


Figure 8. An instance of global-scale reasoning about uncertainty.

4.2. Back-Propagation of Exploration Bonuses

Our propositions will be presented in three steps, each one corresponding to an improvement of the algorithms. We shall be concerned first with ADP; the case of QL will be examined in section 4.2.3.

4.2.1. Basic principle

The fact that IDP– and IEDP– represent only local exploration policies clearly appears in the fundamental equation (30):

$$\begin{aligned}
 N_i^k &\stackrel{def}{=} V_i^k + \delta_i^k \\
 &\simeq \left(\tilde{q}_i^k + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l V_j^l \right) + \delta_i^k \\
 &\simeq (\tilde{q}_i^k + \delta_i^k) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l V_j^l \quad (\forall (i, k)).
 \end{aligned} \tag{37}$$

We see that only the exploration bonus associated with the next action is present. Thus, Σ_i is not motivated in any way by observations that could be made in states other than x_i , for all i .

To take into account the uncertainty attached to states other than the current one, the associated exploration bonuses must be present in the criterion. If we use, for instance

$$N_i^k \stackrel{def}{=} (\tilde{q}_i^k + \delta_i^k) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l (V_j^l + \delta_j^l) \quad (\forall i, k), \tag{38}$$

then the next two observations are taken into account. The exploration bonus of the second action is discounted as a reward. When $i = j$ and $k = l$, the bonus of the second action is over-evaluated because the decreasing of δ_i^k after the transition (i, j, k) is neglected: $\delta_j^l = \delta_i^k = \delta(n_i^k)$ is used instead of $\delta(n_i^k + 1)$.

If we accept approximations of this kind, it is possible to take into account the uncertainty attached to every state that we may reach in the future. To do this, we will try to find the solution of Bellman's system defined by

$$\nu_i^k = (\tilde{q}_i^k + \delta_i^k) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l \nu_j^l \quad (\forall (i, k)), \tag{39}$$

using, for instance, an asynchronous DP algorithm based on the unit operation

$$N_i^k \leftarrow (\tilde{q}_i^k + \delta_i^k) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l N_j^l. \tag{40}$$

In this way, the exploration bonus of every future action is represented implicitly in the variables N_i^k . The bonuses are discounted at the same rate as the rewards. As

in (38), the decreasing of $\delta_0(n)$ with n is neglected, and thus most of the bonuses are over-evaluated⁷.

Despite this approximation, an algorithm based on (40) is a global-scale exploration strategy: each decision takes into account the uncertainty in each state attainable in the future, and the observation of a_k in x_i implies an update of δ_i^k , which may cause a change of the decision in some state x_j ($j \neq i$). This constitutes an approximate solution for two reasons:

1. the uncertainty is measured with formulas valid for normal bandit problems,
2. the decrease of $\delta_0(n)$ is neglected.

Therefore, the fundamental principle of the algorithms developed here is to *add the exploration bonuses to rewards and to introduce them into the DP process*. They are then back-propagated through the state lattice to simulate global-scale reasoning about uncertainty.

4.2.2. Scaling of the exploration bonuses

We have stressed in section 3.1.4 the importance of the order of magnitude of the exploration bonuses in the different solutions of bandit problems: the optimal behavior defined as $\max_k (\bar{\rho}_k + \delta_k)$ may be significantly changed if we multiply all the bonuses δ_k by a positive constant.

The Bayesian and non-Bayesian statistics propose to calculate the exploration bonus δ_k starting from a unit exploration bonus $\delta_0(n_k)$, whose value varies between rather close bounds as a function of the external parameter θ or g . The unit bonus is then scaled by the standard deviation of ρ_k (measured or not), and thus, the resulting bonus δ_k may be considered as proportionate to $\bar{\rho}_k$.

The algorithms developed in the previous section keep close to these propositions, because the criterion N_i^k defined by (30) is an average cumulated reward V_i^k , plus an exploration bonus δ_i^k proportionate to this quantity, by the presence of the standard deviation s_i^k or σ_{max} in its definition (28) or (36). Informally, we say that (30) has the form

$$V + \delta_V = \left(\sum_{t=0}^{\infty} \gamma^t \cdot r \right) + \delta_V, \quad (41)$$

where V a mean V-value, and δ_V and r represent, respectively, an exploration bonus proportionate to V and a mean reward.

With the same notation, the order of magnitude of the criterion (40) is

$$\sum_{t=0}^{\infty} \gamma^t (r + \delta_V) = V + \frac{\delta_V}{1 - \gamma}. \quad (42)$$

We see that the bonus is multiplied by the constant $(1 - \gamma)^{-1} > 1$. Because γ is usually close to 1, $(1 - \gamma)^{-1}$ is usually a great positive value. Therefore, the algorithms based on (40) explore much more than the original algorithms.

This problem is easily solved by multiplying all the exploration bonuses by $(1 - \gamma)$ before introducing them into the DP process. The fundamental equation (39) then takes its final shape

$$\nu_i^k = (\tilde{q}_i^k + \delta_i^k (1 - \gamma)) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l \nu_j^l \quad (\forall (i, k)), \quad (43)$$

and the associated DP equation is

$$N_i^k \leftarrow (\tilde{q}_i^k + \delta_i^k (1 - \gamma)) + \gamma \sum_j \tilde{p}_{ij}^k \cdot \max_l N_j^l. \quad (44)$$

The criterion is thus brought back to the order of magnitude (41), where δ_V is proportionate to expected cumulated rewards V , but it is composed at $\gamma^t (1 - \gamma)$ 100% of the bonus associated with the action executed at time t , for all t .

Scaling the exploration bonuses to take into account the discounting is the second technique used in the algorithms presented in this section. Figure 9 describes their operation in comparison with ‘-’ algorithms.

4.2.3. Initialization of variables

Equation (44) allows the derivation of efficient indirect algorithms. Now we try to apply the ideas developed above to QL.

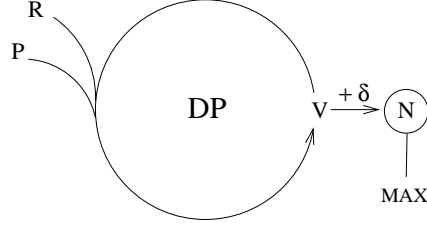
The QL equation associated with (44) is

$$\Delta N_i^k = \alpha (n_i^k) \left[(r_{ij}^k + \delta_i^k (1 - \gamma)) + \gamma \cdot \max_l N_j^l - N_i^k \right]. \quad (45)$$

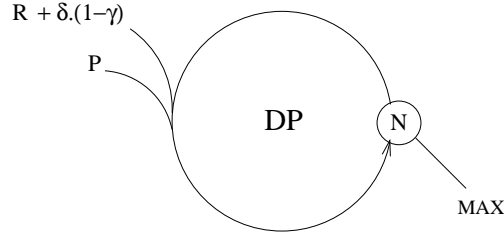
This operation is executed each time that transition (i, j, k) occurs, and only at these times. This is the main limitation of QL: it cannot back-propagate the value of a state-action pair without passing through this state and executing this action. Therefore, it must travel through the problem not only for the needs of exploration, but also to calculate the estimated optimal policy, i.e. for the needs of exploitation. This weakness has been illustrated by the experiments of Barto et al. (1991, 1995). It is at the origin of the development of intermediate solutions such as Sutton’s (1990, 1991b) Dyna architecture.

In the case of our “N-learning”, this drawback may be disastrous because the bonus δ_i^k varies at each transition (i, j, k) , for all j . The MDP implicitly defined by (43) changes at each time step, i.e. much faster than QL solves it. Thus, it appears that a model of the problem is required for implementing the back-propagation of exploration bonuses in a perfect manner.

However, we think that it is possible to have at least an approximate mechanism of back-propagation of uncertainty in a direct algorithm such as QL. We argue that,



9a. Local exploration (equation (30)): DP calculates the V-values in an iterative way, starting from the rewards R and the estimated probabilities \tilde{P} (figured as P). When the V-values are “ready”, the exploration bonuses δ are added to them to get the criterion N that is maximized at each time step.



9b. Global exploration (equation (44)): the exploration bonuses are scaled and added to the rewards before starting the DP iterations. Hereafter, DP directly calculates the variables N that are maximized at each time step.

Figure 9. Functioning of bandit problem-based algorithms.

even if it is not rigorously stated, equation (45) deserves to be tried. However, some preliminary considerations about the initial values of the variables are necessary.

Theoretically, the initial Q-values do not matter for QL’s convergence, the main requirement being that every action is tried infinitely often in each state. In practice, the Q_i^k are often initialized to 0, because we suppose some symmetry of the problem.

Here, we try to manage the impossibility of an infinite number of trials of each state-action pair. The N-values are made to reflect considerations about uncertainty and inaccuracy, and the actions that maximize these variables are executed exclusively. Therefore, their initial value is very important. One can easily see that setting them to 0 is not satisfying at all: because uncertainty is great in the beginning of the experience, a positive and rather high initial value is better.

Observing that:

1. if the Q-values are initialized to 0, only the part due to uncertainty is present in the initial value of N_i^k ,

2. in the worst-case algorithms, each action a_k is tried at least once in each state x_i , and then the associated exploration bonus δ_i^k is initialized to its value corresponding to one observation:

$$\delta_1 \stackrel{def}{=} \sigma_{max} \cdot \delta_0(1) = \sigma_{max} \begin{pmatrix} z_{\alpha/2} \\ \text{or} \\ \nu_g(0, 1) \end{pmatrix}, \quad (46)$$

3. in the variance- and error-based algorithms, each action is tried twice in each state, and the initial value of δ_i^k varies as a function of s_i^k , for all (i, k) ;

we propose to initialize the N_i^k to the value δ_1 defined

- by equation (46) in the worst-case algorithms,
- as an external parameter in the case of variance- and error-based algorithms. Because this parameter represents the exploration bonus associated to an action that has been tried only once, one must assign to it a value greater than any possible exploration bonus. For instance, if the difference $r_M - r_m$ is known, one could choose $\delta_1 > \sigma_{max} \cdot \delta_0(2)$. Moreover, if one wants to preserve the convexity of the exploration bonus with respect to n (see fig. 2 and 3), one should choose $\delta_1 > \sigma_{max} (2 \cdot \delta_0(2) - \delta_0(3))$.

With its main variables initialized to this rather high value (because of the presence of the factor $(1 - \gamma)^{-1}$ in the definition (35) of σ_{max}), the algorithm is forced to explore the environment completely. Actually, the simulation results presented in the next section show very positive results of a system based on (45) if we introduce this later sophistication.

Therefore, the last technique used in the algorithms presented in this section is *to initialize the main variables with the value of an exploration bonus after one observation, or with another large positive constant.*

This is an instance of the heuristic called “optimism in the face of uncertainty” (Kaelbling et al. 1995) already used by several authors to favor exploration (Schmidhuber 1991, Moore and Atkeson 1993, Kaelbling 1993, Koenig and Simmons 1996). Its use is perfectly justified in our framework, because the main variables reflect the uncertainty which is high in the beginning of experiments.

This heuristic is useless in the case of ADP, as long as the DP stage between two decision-times is performed synchronously. In this case, using it is equivalent to adding the same positive constant to each N_i^k at each time t . However, this heuristic is necessary when we apply the ideas developed above to any asynchronous DP-based algorithm which executes back-ups of some state-action pairs before having tried all of them once. This is true for QL as well as for Barto et al. ARTDP algorithm (1991, 1995) and for Sutton’s Dyna architectures (1990, 1991b).

- 1. Initialize:**
 - set counters to 0: $n_{ij}^k = n_i^k = 0$ for all (i, j, k) ,
 - initialize the sliding window of each state action pair (i, k) ,
 - initialize the usual main variables: $V_i^k = 0$ (ADP) or $Q_i^k = 0$ (QL) for all (i, k) ,
 - initialize the exploration bonuses: $\delta_i^k = \delta_1$ for all (i, k) ,
 - initialize the special main variables: $N_i^k = \delta_1$ for all (i, k) .
- 2. Choose an action** (the current state is x_i):
 - if there exists k' such that $n_i^{k'} < 2$, then choose $k = k'$,
 - otherwise choose $k = \arg \max_{k'} N_i^{k'}$.
- 3. Execute action** a_k , observe arrival state x_j .
- 4. Update variables:**
 - increment the counters n_i^k and n_{ij}^k ,
 - update the estimate \tilde{p}_{ij}^k with (7), if using ADP,
 - update the usual main variables:
 - ADP: update $V_{i'}^{k'}$ with (8) and (9) for different (i', k') such that $n_{i'}^{k'} > 0$,
 - QL: update Q_i^k with (10),
 - add an observation to the sliding window of state-action pair (i, k) with (26) (variance-based) or (32) (error-based),
 - if $n_i^k > 1$:
 - update the statistics $\bar{\rho}_i^k$ and s_i^k with (14) and (16),
 - update the exploration bonus δ_i^k with (28).
 - update the special main variables:
 - ADP: update $N_{i'}^{k'}$ with (44) and (9) for different (i', k') such that $n_{i'}^{k'} > 0$,
 - QL: update N_i^k with (45),
- 5. Increment time**, return to 2.

Figure 10. Outline of “+” algorithms’ variance- and error-based variants (ADP and QL).

4.3. Algorithms

Using the same nomenclature as in section 3, we call variance- or error-based IDP+, IEDP+, IQL+, and IEQL+ the algorithms defined by the following set of formulas:

- the equations of ADP or QL: (7), (8) and (9); or (10) and (11);
- the equation that is chosen for the reward ρ_i^k introduced in the sliding windows: (26) or (32);
- the definition of the statistics $\bar{\rho}_i^k$ and s_i^k : (14) and (16);
- the definition of the exploration bonus: (28);
- the special ADP or QL equation: (44) or (45).

These algorithms are depicted in fig. 10.

These algorithms use two sets of variables : the V_i^k or Q_i^k , and the N_i^k . All the operations usually executed on the first (V or Q) are also applied to the second (N). The first are used to calculate the sample standard deviation s_i^k , the second

- 1. Initialize:**
 - set the counters to 0: $n_{ij}^k = n_i^k = 0$ for all (i, j, k) ,
 - initialize the main variables: $N_i^k = \sigma_{max} \cdot \delta_0(1)$ for all (i, k) .
 - 2. Choose an action**(the current state is x_i):
 - if there exists k' such that $n_i^{k'} = 0$, then choose $k = k'$,
 - otherwise choose $k = \arg \max_{k'} N_i^{k'}$.
 - 3. Execute action** a_k , observe arrival state x_j .
 - 4. Update variables:**
 - increment counters n_i^k and n_{ij}^k ,
 - update the estimate \hat{p}_{ij}^k with (7), if using ADP,
 - update the exploration bonus δ_i^k with (36),
 - update main variables:
 - ADP: update $N_{i'}^{k'}$ with (44) and (9) for different (i', k') such that $n_{i'}^{k'} > 0$,
 - QL: update N_i^k with (45).
 - 5. Increment time**, return to 2.

Figure 11. Outline of “+” algorithms’ worst-case variant (ADP and QL).

are the main variables of the algorithms. Because of this double set of variables, the time of calculation is multiplied by two⁸.

Worst-case IDP+, IEDP+, IQL+, and IEQL+ do not use two sets of variables, because the standard deviations are not measured but are taken at their greatest possible value. These variants use the following formulas:

- the definition of the exploration bonus: (36);
- the special ADP or QL equations: (7), (9) and (44), or (11) and (45).

They are represented in fig. 11. It is important to be note that their complexity is very close to the complexity of the original algorithm (ADP or QL) with greedy exploitation.

4.4. Other Instances of Back-Propagation of Uncertainty

The back-propagation of uncertainty has already been used by Sutton, and it may be implemented in every algorithm that uses an exploration bonus. Moreover, a similar mechanism may be used to direct the first step of the exploration of synchronous ADP.

4.4.1. Sutton’s Dyna-Q+

In a paper about the applications of Dyna architectures to QL, Sutton (1990) defines two algorithms:

- Dyna-Q– that uses the usual QL equation (10) and always chooses the action that maximizes the recency-based criterion

$$Q_i^k + \varepsilon \sqrt{\Delta_i^k}, \quad (47)$$

where Δ_i^k is the time elapsed since the last try of the state-action pair (i, k) , and $\varepsilon > 0$ is an external parameter;

Dyna-Q+ that does not use (10) but

$$\Delta Q_i^k = \alpha(n_i^k) \left[\left(r_{ij}^k + \varepsilon \sqrt{\Delta_i^k} \right) + \gamma \cdot \max_l Q_j^l - Q_i^k \right], \quad (48)$$

and always chooses the actions that maximize the Q-values calculated in this way.

Equation (48) is very close to (45), the main difference being the way in which the exploration bonus is defined: the stationary bandit problem's bonus (28) or (36) is replaced by Sutton's recency-based exploration bonus that is adapted to non-stationary environments. Note that:

- it is useless to scale the exploration bonuses by the factor $(1 - \gamma)$, because Sutton's bonuses are directly proportional to the external parameter ε , thus changing ε for $\varepsilon(1 - \gamma)$ has the same effect;
- Sutton says to “simulate state transitions that have never occurred”. This results in back-propagating the bonuses of actions that have never been tried before, and thus this is an alternative to the initialization of N_i^k with large positive values.

Thus, Sutton already used the back-propagation of exploration bonuses. However, our work is innovative in the following two ways:

1. We have proposed algorithms for stationary MDPs with adapted definitions of the exploration bonuses.
2. Although it is derived from QL, Dyna-Q+ uses a model of the unknown process. Thus, this is not really a direct algorithm, and one could think that the principle of back-propagating uncertainty could not be made to work in direct algorithms (cf. e.g. Thrun 1992a). However, IQL+ and IEQL+ are completely direct algorithms that successfully implement this principle (see section 5).

Like ours, Sutton's experiments show an important improvement of the algorithm's performance due to back-propagation of exploration bonuses. This constitutes a convincing empirical demonstration of the utility of this mechanism.

4.4.2. Directed exploration

It is possible to introduce the back-propagation of uncertainty in every algorithm that uses exploration bonuses, as we did in section 4.2. For instance, the different directed techniques presented by Thrun (1992a) use a criterion of the form (31). Therefore, we can build a ‘+’ version of each one by introducing the back-propagation of exploration bonuses. Note that:

- the scaling of the exploration bonuses is useless when the bonuses are directly proportional to an external parameter, as are Sutton’s and most of Thrun’s bonuses. In this case multiplying the external parameter by $(1 - \gamma)$ has the same effect;
- as stated in 4.2.3, the initialization of the main variables with a positive constant is unnecessary in synchronous ADP, since this is equivalent to adding a positive constant to each variable at each time. However, it should be used in asynchronous algorithms such as ARTDP and QL.

4.4.3. The first steps of ADP

Synchronous and Gauss-Seidel versions of ADP are characterized by the fact that each state-action pair (i, k) is backed up at least once during each DP stage, between two decision-times (cf. Barto et al. 1991, 1995). Thus, these algorithms cannot calculate the value function before the MLE (7) is completely defined, i.e. each action has been tried at least once in each state ($n_i^k > 0$, for all (i, k)). To achieve this, one will force the algorithm to choose an action that has never been tried in the current state, each time that such an action is available (i.e. the algorithms first try each action once in each state). However, this does not define a complete policy, since we have not decided what to do when the algorithm returns in a state where every action has already been tried, while the MLE is still incomplete.

A completely defined strategy is, for instance: in each state, try every action in an arbitrary order and in a cyclical way, until the MLE is complete. This is equivalent to always choosing the actions that minimize the counter n_i^k , the index k determining the order in which actions are selected. Therefore, the counters n_i^k represent a kind of “exploration penalty” that must be minimized for exploration. In regard to this penalty, the algorithm works in a greedy way, i.e. it does not anticipate penalties other than the immediate. For this reason, the algorithm constitutes a local exploration policy.

As in section 4.2, we can deduce from this local criterion a global exploration policy. It consists of always choosing the actions that minimize the variable ξ_i^k defined by

$$\xi_i^k = \begin{cases} n_i^k + \gamma \sum_j \tilde{p}_{ij}^k \cdot \min_l \xi_j^l & \text{if } n_i^k > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (\forall (i, k)). \quad (49)$$

These equations can be solved using an asynchronous DP algorithm that does not back-up the state-action pairs (i, k) such that $n_i^k = 0$.

5. Numerical Simulations

Extensive numerical simulations have been run to test the algorithms presented in this paper and to compare them with some well-known techniques for action selection. In general, these experiments show that very good performance is attained by the ‘+’ algorithms, particularly their worst-case variants. A representative subset of these results is reproduced here.

5.1. Experimental Protocol

The protocol that we followed is very close to Kaelbling’s (1993, chap. 9). First, we implemented a selection of algorithms including our own. Then, we set up a benchmark of several small MDPs taken from previous material, or that we built to cover the set of possible problems as widely as possible. Finally, we ran each algorithm on each problem, hiding the transition probabilities so that the algorithm must discover them on-line. During these experiments, we measured the evolution of the rewards received by the algorithms as a function of time and of other performance criteria. Here, we present the results obtained with 3 environments of our benchmark. More results are available in Meuleau’s thesis (1996).

5.1.1. Algorithms

The results we present were obtained with the following 16 exploration techniques:

- two undirected techniques: the semi-uniform distribution and Boltzmann law, with constant parameters P_{best} and T (cf. Thrun 1992a, 1992b),
- a selection of directed techniques:
 - Sato et al.’s (1988) asymptotically optimal technique,
 - Sutton’s (1990, 1991a) recency-based technique, implemented in its local form as in Dyna-Q– (see section 4.4.1),
 - our 12 proposed techniques: variance-based, error-based and worst-case I and IE algorithms, implemented in their local (–) or global (+) forms.

Each technique was implemented both in QL and in a version of ADP that executes two iterations of Gauss-Seidel DP after each state transition⁹. This constitutes a total of 32 different algorithms.

5.1.2. Environments

The results presented here concern three stochastic MDPs used as testing environments.

Tasks 1 and 2 are derived from the deterministic MDPs presented in figs. 6 and 7 respectively (section 3.5), by including 20% inherent randomness: each time that an action is chosen by an algorithm, there is a 0.2 probability that the other action is executed, without the system's being aware of this error. This probability value was chosen because it leaves the optimal policy unchanged.

These two environments were built to mislead local exploration policies. As we will see, it appeared during the simulations that task 1 represents a very difficult problem for these techniques, whereas task 2 is less misleading.

Task 3 is a five-state, three-action MDP taken from Sato et al.'s (1988) paper. Because it is characterized by a positive transition matrix ($p_{ij}^k > 0$ for all (i, k)), this problem should not favor global exploration policies.

Therefore, the three environments used in this paper are of decreasing difficulty for local exploration policies. Note that task 2 presents a sparse reward matrix and thus it may mislead variance- and error-based algorithms.

5.1.3. Measures

An experiment consists of testing an algorithm in an environment, after having fixed the algorithm parameters. Each experiment has a fixed duration of 5000 time-steps. The initial state of the system is x_0 in the case of tasks 1 and 2, and is drawn at random and uniformly in the case of task 3.

The performance of an algorithm during an experiment is defined as the average reward that it receives per time-step. It is interesting to compare this measure with the optimal performance, i.e. the expected rate of reward if one always executes the optimal policy defined by (5) (which is reachable only if the problem is known in advance). Howard (1960) has developed a technique for calculating this data. The results of these calculations will be presented with the simulation results.

Note that the "optimal performance" defined above is not the greatest expected performance possible. Instead, the greatest expected performance is realized by the policy that maximizes a 5000-step horizon criterion

$$\mathbb{E} \left(\sum_{t=0}^{4999} r(t) \right), \quad (50)$$

which usually differs from policy (5). This suggests using the discounted cumulated reward received during the experiment as performance criterion, so that the policy that realizes the greatest expected performance is very close to the optimal policy (5).¹⁰ However, early experiments showed that this criterion does not establish any discrimination between the different algorithms, as it places too much emphasis on the first steps of the experiments when all the algorithms are equally as inefficient.

Another measure used is the rate of convergence of the algorithms over several experiments in the same environment. This is the percentage of times that the estimated optimal policy is precisely the optimal policy (5) at the end of the run.

The last measure recorded is the learning curve. This is the graph showing the average reward received at time t , as a function of t (averages are calculated over several experiments in the same environment). Because these raw data are always very noisy, they have to be smoothed by moving averages.

5.1.4. Protocol

Two series of experiments are executed for each algorithm and each environment:

1. a first set of experiments allows for optimizing the algorithm parameters. This is done by looking at the evolution of the average performance over 100 experiments, as a function of these parameters. See Meuleau (1996) for more details about parameter optimization;
2. a second set of 1000 experiments is conducted with the parameters set to their optimal value. Then we measure the rate of convergence and draw the learning curves.

Note that only the best variants (variance-based, error-based, or worst-case) of our algorithms were submitted to the second series of experiments.

5.2. Simulation Results and Comments

Simulation results are presented in tables 1 to 3, and in figures 12 to 14.

5.2.1. Task 1

Task 1 (table 1, fig. 12) is the most discriminating of the three problems. Results in this environment clearly show the superiority of directed exploration techniques over undirected, and of global exploration over local. This classification is striking in the case of QL, and less apparent, but still present, in the case of ADP.

The observed oscillating behavior of Sutton’s recency-based QL is characteristic of this exploration technique: first, the algorithm finds the local optimum defined as always choosing the action a_0 . As time passes, the algorithm is pushed by the recency-based exploration bonuses to explore more profoundly. Then it finds the optimal policy. However, the algorithm will never converge on the optimal policy, but instead will regularly restart exploration. This behavior is adapted to non-stationary environments. In stationary environments, it allows a good convergence rate, but poor performance.

Table 1. Simulation results for task 1: performance of the algorithms (standard deviation), and rate of convergence (conv.) of the best variant (vb.: variance-based, eb.: error-based, wc.: worst-case). The expected optimal performance (attained when the problem is known in advance) is 3.02.

Exploration policy	QL	ADP
Semi-uniform distribution	1.60 (0.19), conv. = 46.5 %	2.34 (0.55), conv. = 73.2 %
Boltzmann law	1.53 (0.38), conv. = 58.2 %	2.67 (0.31), conv. = 98.4 %
Sato (asymptotically optimal)	1.82 (0.36), conv. = 28.3 %	2.62 (0.54), conv. = 82.3 %
Sutton (recency-based)	2.28 (0.23), conv. = 99.6 %	2.77 (0.26), conv. = 99.8 %
vb. IEQL- or IEDP-	1.80 (0.47)	2.79 (0.27)
eb. IEQL- or IEDP-	1.80 (0.51)	2.81 (0.23), conv. = 97.1 %
wc. IEQL- or IEDP-	2.00 (0.48), conv. = 22.0 %	2.21 (0.18)
vb. IQL- or IDP-	1.66 (0.55)	2.81 (0.27), conv. = 99.2 %
eb. IQL- or IDP-	1.68 (0.42)	2.80 (0.31)
wc. IQL- or IDP-	2.03 (0.37), conv. = 41.7 %	2.18 (0.19)
vb. IEQL+ or IEDP+	2.78 (0.13)	2.85 (0.28)
eb. IEQL+ or IEDP+	2.61 (0.15)	2.88 (0.25), conv. = 98.5 %
wc. IEQL+ or IEDP+	2.84 (0.16), conv. = 79.7 %	2.88 (0.20), conv. = 95.3 %
vb. IQL+ or IDP+	2.57 (0.15)	2.83 (0.33)
eb. IQL+ or IDP+	2.53 (0.15)	2.83 (0.33)
wc. IQL+ or IDP+	2.85 (0.16), conv. = 88.2 %	2.87 (0.26), conv. = 97.7 %

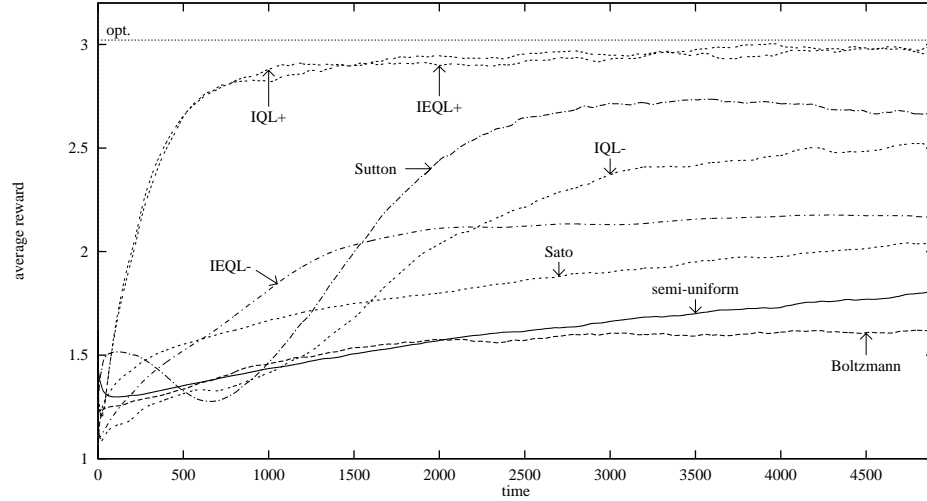


Figure 12a. Learning curves of Q-learning for task 1.

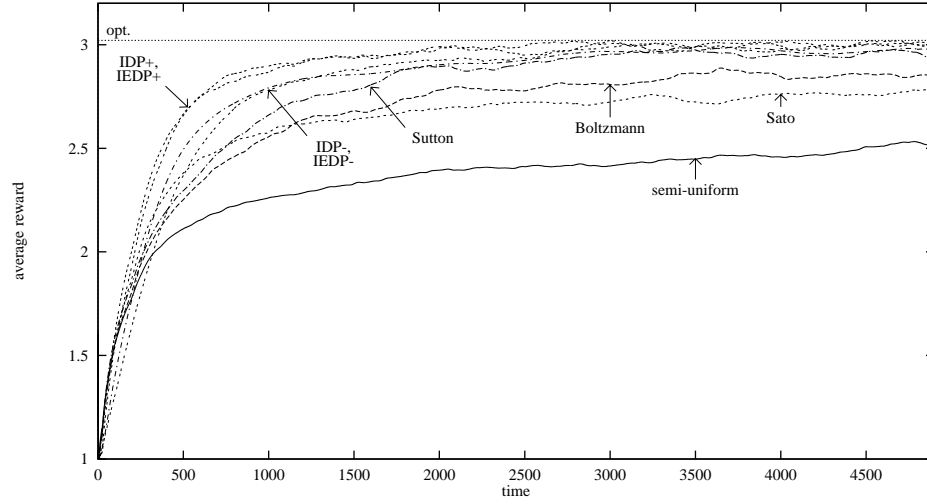
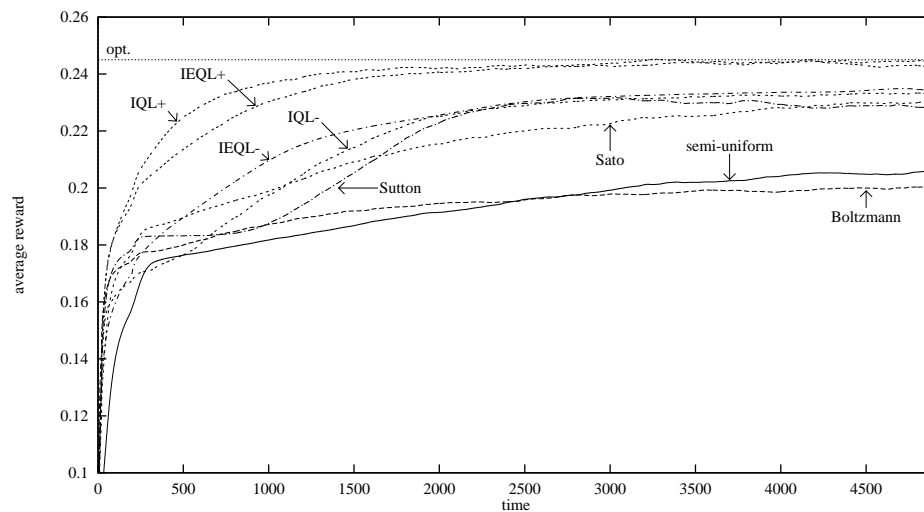


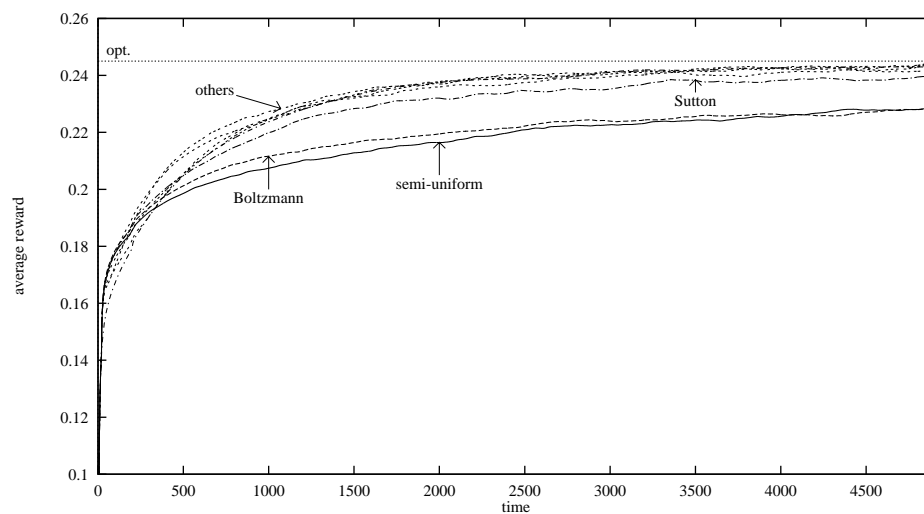
Figure 12b. Learning curves of adaptive dynamic programming for task 1.

Table 2. Simulation results for task 2: performance of the algorithms (standard deviation), and rate of convergence (conv.) of the best variant (vb.: variance-based, eb.: error-based, wc.: worst-case). The expected optimal performance (attained when the problem is known in advance) is 0.245.

Exploration policy	QL	ADP
Semi-uniform distribution	0.192 (0.011), conv. = 91.4 %	0.216 (0.022), conv. = 88.8 %
Boltzmann law	0.193 (0.019), conv. = 88.5 %	0.218 (0.025), conv. = 83.3 %
Sato (asymptotically optimal)	0.214 (0.012), conv. = 96.3 %	0.232 (0.018), conv. = 97.2 %
Sutton (recency-based)	0.215 (0.008), conv. = 100.0 %	0.228 (0.016), conv. = 98.8 %
vb. IEQL- or IEDP-	0.102 (0.028)	0.231 (0.019)
eb. IEQL- or IEDP-	0.104 (0.034)	0.232 (0.016), conv. = 98.3 %
wc. IEQL- or IEDP-	0.221 (0.014), conv. = 99.7 %	0.223 (0.014)
vb. IQL- or IDP-	0.103 (0.034)	0.231 (0.020)
eb. IQL- or IDP-	0.100 (0.025)	0.231 (0.017)
wc. IQL- or IDP-	0.217 (0.015), conv. = 99.9 %	0.232 (0.19), conv. = 98.5 %
vb. IEQL+ or IEDP+	0.179 (0.003)	0.233 (0.017), conv. = 97.6 %
eb. IEQL+ or IEDP+	0.179 (0.004)	0.232 (0.018)
wc. IEQL+ or IEDP+	0.235 (0.008), conv. = 100.0 %	0.228 (0.017)
vb. IQL+ or IDP+	0.179 (0.004)	0.231 (0.020)
eb. IQL+ or IDP+	0.179 (0.004)	0.231 (0.19)
wc. IQL+ or IDP+	0.238 (0.007), conv. = 100.0 %	0.232 (0.017), conv. = 98.0 %



13a. Q-learning.



13b. Adaptive dynamic programming (“others”: Sato, IEDP-, IDP-, IEDP+, and IDP+).

Figure 13. Learning curves for task 2.

5.2.2. Task 2

Task 2 (table 2, fig. 13) is less discriminating. In the case of QL, it highlights the same classification of the techniques as task 1. In the case of ADP, results do not show a superiority of global exploration over local.

Variance- and error-based IQL and IEQL suffer from the degeneration of the exploration bonuses due the sparse reward matrix (cf. 3.4.3). However, this does not happen with indirect algorithms.

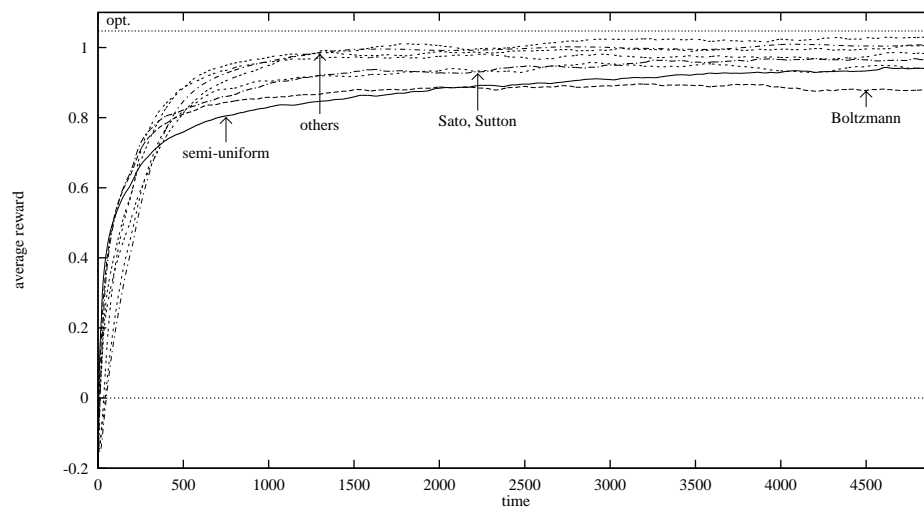
The most surprising fact is that the best performance is attained by direct algorithms: IQL+ and IEQL+ perform better than IDP+ and IDP+. This phenomenon was observed in almost half of the environments of our complete benchmark. Further experiments are needed to understand this result which suggests that a model of the problem is not necessary if the exploration policy is efficient, at least in small environments.

5.2.3. Task 3

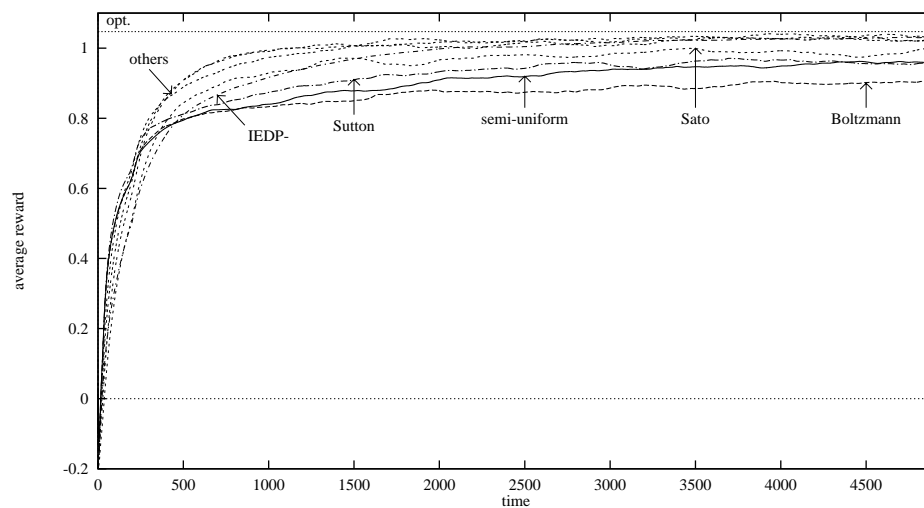
Task 3 (table 3, fig. 14) was chosen because it should not favor global exploration policies. Simulation results fit this expectation rather well: they show a superiority of directed exploration over undirected, but no clear superiority of global

Table 3. Simulation results for task 3: performance of the algorithms (standard deviation), and rate of convergence (conv.) of the best variant (vb.: variance-based, eb.: error-based, wc.: worst-case). The expected optimal performance (attained when the problem is known in advance) is 1.05.

Exploration policy	QL	ADP
Semi-uniform distribution	0.87 (0.11), conv. = 78.4 %	0.89 (0.11), conv. = 80.9 %
Boltzmann law	0.86 (0.13), conv. = 60.6 %	0.85 (0.16), conv. = 95.6 %
Sato (asymptotically optimal)	0.89 (0.14), conv. = 63.2 %	0.93 (0.10), conv. = 78.8 %
Sutton (recency-based)	0.91 (0.08), conv. = 88.9 %	0.91 (0.08), conv. = 96.2 %
vb. IEQL- or IEDP-	0.94 (0.11)	0.95 (0.10), conv. = 84.9 %
eb. IEQL- or IEDP-	0.95 (0.10), conv. = 80.5 %	0.95 (0.12), conv. = 75.1 %
wc. IEQL- or IEDP-	0.93 (0.12)	0.95 (0.05), conv. = 99.7 %
vb. IQL- or IDP-	0.93 (0.13)	0.96 (0.09)
eb. IQL- or IDP-	0.94 (0.12), conv. = 69.4 %	0.96 (0.11)
wc. IQL- or IDP-	0.93 (0.12)	0.98 (0.07), conv. = 94.8 %
vb. IEQL+ or IEDP+	0.71 (0.05)	0.96 (0.09)
eb. IEQL+ or IEDP+	0.71 (0.05)	0.97 (0.10), conv. = 82.5 %
wc. IEQL+ or IEDP+	0.96 (0.08), conv. = 91.2 %	0.97 (0.07), conv. = 92.7 %
vb. IQL+ or IDP+	0.72 (0.05)	0.92 (0.14)
eb. IQL+ or IDP+	0.72 (0.05)	0.91 (0.16)
wc. IQL+ or IDP+	0.95 (0.10), conv. = 78.9 %	0.98 (0.08), conv. = 91.6 %



14a. Q-learning (“others”: IEQL−, IQL−, IEQL+, and IQL+).



14b. Adaptive dynamic programming (“others”: IDP−, IEDP+, and IDP+).

Figure 14. Learning curves for task 3.

exploration over local. However, among the directed techniques tested (local and global), our algorithms behave better than others.

Even if the results do not show a clear superiority of global exploration in this completely connected environment, the ‘+’ algorithms do not perform worse than the ‘-’. This tends to show that, even if the back-propagation of exploration bonuses does not always increase the performance of the algorithms, it does not decrease it.

The surprising result is the bad performance of variance- and error-based IQL+ and IEQL+. Because task 3 has a positive transition matrix and a full (as opposed to sparse) reward matrix, the degeneration of exploration bonuses should not happen. Actually, variance- and error-based ‘-’ algorithms perform well. Further experiments are required to explain why the ‘+’ algorithms fail on this task.

5.2.4. Discussion

Our experiments show the following classification of exploration techniques (by increasing efficiency):

1. undirected exploration,
2. directed, local exploration,
3. directed, global exploration.

These results appear very clearly in the case of QL, especially, but not exclusively, in environments that were designed to be discriminating. Results with ADP are less significant, probably because the environments used are too small.

Among the different variants tested, worst-case algorithms are the simplest and the most efficient. They represent a very good compromise between these two objectives. In particular one should remember worst-case IQL+ and IEQL+ which, despite their low complexity, achieve the best performances of QL in all the environments of our complete benchmark.

6. Conclusion

This paper makes several contribution to the problem of exploration of multi-state environments. First, we proposed a unified notation for different solutions to normal bandit problems, and we stressed the importance of the notion of exploration bonuses (section 3.1). Second, we highlighted several theoretical and practical limitations to the approach that consists of using bandit problem theory to define local measures of the uncertainty in multi-state environments (sections 3.2 and 3.5). Finally, we proposed efficient algorithms built on the basis of these results (section 4).

Qualitatively, the algorithms that we propose may be defended by the following arguments:

1. exploration bonuses are defined to evaluate the interest of the actions in regards to exploration. They allow us to quantify the uncertainty in the same units as the rewards, and to make explicit the reasons for the choice of a non-optimal action;
2. the back-propagation of exploration bonuses allows an intelligent and complete exploration of the environment, using only local measures of uncertainty;
3. the scaling of exploration bonuses is necessary to moderate the effect of the back-propagation of bonuses; it allows a reasonable exploration;
4. the initialization of variables with a large positive constant allows exploration to start from time 0, even if there is no model of the problem. This technique is necessary in asynchronous algorithms to ensure correct behavior before the environment has been completely visited.

Experiments showed that very good performance can be attained if these four techniques are jointly applied. Further research is needed to determine the relative role of each one in the origin of this success.

In the introduction, we said that most reinforcement learning algorithms may be divided into two components, the first being responsible for the calculation and the storage of the value function, and the second being the rule of action selection. It is noteworthy that this distinction is no longer relevant in ‘+’ algorithms: the variables N_i^k calculated by (44) and (45) represent, at the same time, the value function and the rule of selection of the actions. Our work shows that introducing considerations about uncertainty in the value function is an elegant and efficient solution for the exploration of multi-state environments.

Acknowledgments

We wish to thank Leslie P. Kaelbling, Marco Dorigo, Pierre Saurel and the anonymous reviewers of this article for their very valuable comments and suggestions. During this work, Nicolas Meuleau was supported by Cemagref, the French Institute for Agricultural and Environmental Research.

Notes

1. The following results can be generalized to problems where the set of available actions depends on the state.
2. A short discussion on this subject is present at the end of section 3.4.3.
3. We do not use the usual notation α for the confidence coefficient, because we already used this symbol for QL’s learning-rate.
4. We prefer to develop a special notation by replacing rewards r by ρ , and discount factor γ by g , when bandit problems are concerned. This will prevent confusion in the following sections, when we use bandit problem solutions in the framework of MDPs.

5. Gittins provides tables for the quantities $n(1-g)^{1/2}\nu_g(0, n)$ and $\nu_g(0, 1, n)/\nu_g(0, n) - 1$. This is sufficient to calculate $\nu_g(0, n)$ and $\nu_g(0, 1, n)$.
6. This phenomenon causes the degeneration of undirected exploration that turns to random walk, see section 4.2.3.
7. This drawback may be avoided in Barto et al.'s (1991, 1995) ARTDP algorithm.
8. To avoid this, we tried to suppress the V- or Q-values and use only the N-values, even for feeding the sliding windows (N replaces V in (26) and (32)); see Meuleau (1996).
9. Because variance- and error-based IDP+ and IEDP+ have to manage two sets of variables (cf. 4.3), they execute only one iteration of Gauss-Seidel DP with each set of variables after each state transition.
10. Boundary effects due to the finite duration of the experiments introduce few differences between these two policies. However, since their divergence occurs only during the last steps of the experiment, its overall influence is negligible.

References

- Bar-Shalom, Y. (1981). Stochastic dynamic programming: Caution and probing. *IEEE Trans. Automat. Control*, 26, 1184-1195.
- Barto, A. G., Bradtke, S. J., & Singh, S. P. (1991). *Real-time learning and control using asynchronous dynamic programming*. Technical Report 91-57, University of Massachusetts, Dept. of Computer Science.
- Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72, 81-138.
- Berry, D. A., & Fristedt, B. (1985). *Bandit problems: Sequential allocation of experiments*. London, England: Chapman and Hall.
- Bertsekas, D. (1982). Distributed dynamic programming. *IEEE Trans. Automat. Control*, 27, 610-616.
- Feldbaum, A. (1965). *Optimal control systems*. New-York, NY: Academic Press.
- Fiechter, C. N. (1994). Efficient reinforcement learning. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, 88-97. New Brunswick, NJ.
- Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. New-York, NY: John Wiley and sons.
- Howard, R. A. (1960). *Dynamic programming and Markov processes*. Cambridge, MA: MIT Press.
- Kaelbling, L. P. (1993). *Learning in embedded systems*. Cambridge, MA: MIT Press.
- Kaelbling, L. P. (1996). Introduction to *Machine Learning*, 22, 7-9.
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4.
- Kumar, P. R. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23, 329-380.
- Kumar, P. R., & Becker, A. (1982). A new family of optimal adaptive controllers for Markov chains. *IEEE Trans. Automat. Control*, 27, 137-145.
- Kumar, P. R., & Lin, W. (1982). Optimal adaptive controllers for unknown Markov chains. *IEEE Trans. Automat. Control*, 27, 765-774.
- Koenig, S., & Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration. *Machine Learning*, 22, 227-250.
- Larsen, R. J., & Marx, M. L. (1986). *An introduction to mathematical statistics and its applications*. Englewood Cliffs, NJ: Prentice-Hall.
- Mahadevan, S., & Kaelbling, L. P. (1996). The NSF workshop on reinforcement learning: summary and observations. *AI Magazine*, 17, 89-93.
- Martin, J. J. (1967). *Bayesian decision problems and Markov chains*. New-York, NY: John Wiley and sons.

- Meuleau, N. (1996). *Le dilemme entre exploration et exploitation dans l'apprentissage par renforcement : optimisation adaptative des modèles de décision multi-états*. PhD thesis, Université de Caen, Caen, France.
- Moore, A. W. (1990). *Efficient memory-based learning for robot control*. PhD thesis, Trinity Hall, University of Cambridge, England.
- Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13, 103-130.
- Puterman, M. L. (1994). *Markov decision processes*. New-York, NY: John Wiley and sons.
- Sato, M., Abe, K., & Takeda, H. (1982). Learning control of finite Markov chains with unknown transition probabilities. *IEEE Trans. Automat. Control* 27, 502-505.
- Sato, M., Abe, K., & Takeda, H. (1985). An asymptotically optimal controller for finite Markov chains with unknown transition probabilities. *IEEE Trans. Automat. Control*, 30, 1147-1149.
- Sato, M., Abe, K., & Takeda, H. (1988). Learning control of finite Markov chains with an explicit trade-off between estimation and control. *IEEE Trans. Systems, Man and Cybernetics*, 18, 677-684.
- Sato, M., Abe, K., & Takeda, H. (1990). Learning control of finite Markov chains with an explicit trade-off between estimation and control. In D. S. Touretsky et al. (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*, 287-300. San Mateo, CA: Morgan Kaufmann.
- Schmidhuber, J. (1991a). *Adaptive confidence and adaptive curiosity*. Technical Report FKI-149-91, Technische Universität München, München, Germany.
- Schmidhuber, J. (1991b). Curious model-building control systems. In *Proceedings of International Joint Conference on Neural Networks*, 2, 1458-1463. IEEE.
- Snedecor, G. W. & Cochran, G. (1989). *Statistical Methods*. New-York, NY: Mac Graw-Hill.
- Sutton, R. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*, 216-224. San-Mateo, CA: Morgan Kaufmann.
- Sutton, R. (1991a). Integrated modeling and control based on reinforcement learning and dynamic programming. In R. P. Lippmann et al. (Eds.), *Advances in Neural Information Processing 3*, 471-478. San-Mateo, CA: Morgan Kaufmann.
- Sutton, R. (1991b). Dyna, an integrated architecture for learning, planning and reacting. *SIGART Bulletin*, 2, 160-163.
- Sutton, R. (1992). The challenge of reinforcement learning. Introduction to *Machine Learning*, 8, 1-3.
- Sutton, R., Barto, A. G. & Williams, R. J. (1991). Reinforcement learning is direct adaptive optimal control. *Proceedings of the 1991 American Control Conference*, 2143-2146. Boston, MA.
- Thrun, S. (1992a). *Efficient exploration in reinforcement learning*. Technical Report CS-92-102, Carnegie Mellon University, Pittsburgh, PA.
- Thrun, S. (1992b). *The role of exploration in learning control*. In D. A. White and D. A. Sofge (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. NY: Van Nostrand Reinhold.
- Thrun, S., & Möller, K. (1991). *On planning and exploration in non-discrete environments*. Technical Report 528, GMD, Sankt Augustin, Germany.
- Thrun, S., & Möller, K. (1992). Active exploration in dynamic environments. In J. E. Moody et al. (Eds.), *Advances in Neural Information Processing 4*. San-Mateo, CA: Morgan Kaufmann.
- Tsitsiklis, J. N. (1994). Asynchronous stochastic approximations and Q-learning. *Machine Learning*, 16, 185-202.
- Watkins, C. (1989). *Learning from delayed reward*. PhD thesis, University of Cambridge, Cambridge, England.
- Watkins, C. & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8, 279-292.
- Whitehead, S. D., & Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7, 45-83.

Received Date

Accepted Date

Final Manuscript Date